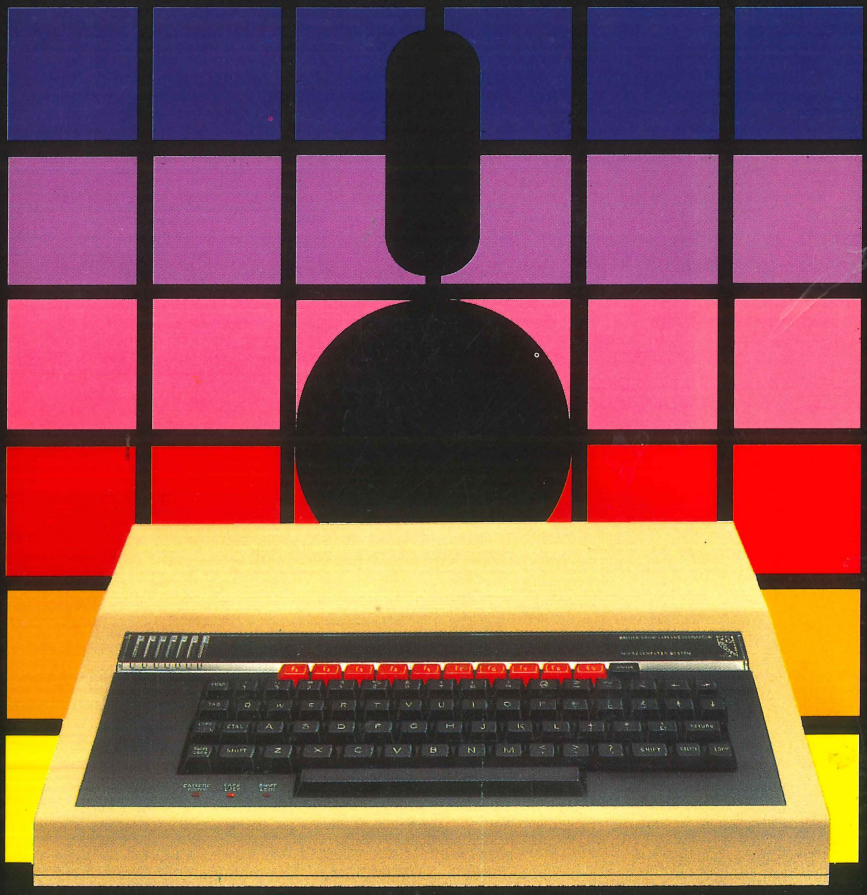
BRITISH BROADCASTING CORPORATION

MICROCOMPUTER SYSTEM

Disc Filing System
USER GUIDE



BBC Microcomputer
Disc Filing System User Guide

Addendum 1

This addendum details the differences between the Acorn 1770 Disc Filing System (DFS) and DFS 1.20 as described in the *Disc Filing System User Guide* Issue 2.

General

The Acorn 1770 DFS (DFS 2.10) is designed to be compatible with the Acorn DFS version 1.20 (and earlier versions). DFS 2.10 works with a 1770 disc controller whereas DFS 1.20 operates with an 8271 disc controller. Programs/ROMs which directly access an 8271 disc controller will not work with a DFS 2.10 disc system. Most discs that work with DFS 1.20 will work with DFS 2.10, obvious exceptions being protected discs which employ direct access of the 8271 disc controller.

DFS 2.10 uses the same commands as DFS 1.20, but some commands have been slightly modifed, as detailed below. There is no utilities disc with DFS 2.10 - the ***FORM** <40/80> and ***VERIFY** utilities which are disc-based with DFS 1.20 are filing system commands with DFS 2.10; references to a utilities disc which occur within the user guide should be ignored. DFS 2.10 contains several new commands which are not present in DFS 1.20 - see below.

Throughout this text references to ROM (read only memory) should be understood to include any non-volatile memory device with ROM compatible connections and access time, such as a 2764, 27128 or 27256 EPROM with 250ns access time.

DFS 2.10-only commands

*CLOSE

Purpose

Closes all sequential files. All disc file data held in RAM output buffers will be copied to the disc. ***CLOSE** has the same effect as the **CLOSE#0** in BASIC.

Example

```
*CLOSE
```

Notes

The minimum abbreviation for ***CLOSE** is ***CL.** .

*EX (<dir>)

Purpose

To display information about the specified directory. The information is displayed across the screen (in hexadecimal) in the following order (reading from left to right): <directory name>.<file name>, protection attribute, load address, execution address, length in bytes, start sector number.

Examples

```
*EX $
```

might give

```
$.!BOOT     L  FF1900 FF8023 00003B 205
$.MENU      L  FF1900 FF1900 000332 008
```

```
*EX D
```

might give

```
D.ROB          FF1900 FF8023 000695 07B
D.WORK         000000 000000 000015 2A9
```

(If no <dir> is given, the currently selected directory is examined.)

Associated commands

*CAT
*INFO

*FORM <40/80> (<drive>) (<drive>) (<drive>) (<drive>).....etc

Purpose

This command is used to format a disc, enabling it to be used with DFS 2.10 (or with earlier DFS versions). The command has the same function as the utilities described in chapter 7 of this user guide, but has slight operating differences as detailed below.

Examples

```
*FORM 40
Format which drive ? 0
GO (Y/N) ? Y
Formatting drive 0 track <track number> (runs from &00 to &27)
```

Formats a 40-track disc in drive 0.

```
*FORM 80
Format which drive ? 0
GO (Y/N) ? Y
Formatting drive 0 track <track number> (runs from &00 to &4F)
```

Formats an 80-track disc in drive 0.

```
*FORM 80 0 2
GO (Y/N) ? Y
Formatting drive 0 track <track number> (runs from &00 to &4F)
Formatting drive 2 track <track number> (runs from &00 to &4F)
```

Formats both sides of an 80-track disc in drive 0.

```
*FORM 80 0 2 1 3
GO (Y/N) ? Y
Formatting drive 0 track <track number> (runs from &00 to &4F)
Formatting drive 2 track <track number> (runs from &00 to &4F)
Formatting drive 1 track <track number> (runs from &00 to &4F)
Formatting drive 3 track <track number> (runs from &00 to &4F)
```

Formats both sides of 80-track discs in drives 0 and 1 of a dual disc drive.

Associated commands

*ENABLE
*VERIFY

Notes

It is not possible to format discs for a 40-track drive in an 80-track drive, nor is it possible to format discs for an 80-track drive in a 40-track drive. (40-track drives use 48 tracks per inch; 80-track drives use 96 tracks per inch.)

The formatting process uses some I/O processor RAM space, (approximately 3.5K bytes above OSHWM (usually **PAGE** in BASIC)), so if you wish to save a program you should do so before formatting a disc. When formatting a disc, screen memory corruption may occur in any non-shadow display mode for values of OSHWM greater than &2200. For example, formatting a disc using a BBC Microcomputer with DFS 2.10 and ADFS installed, and with a second processor connected and switched on, will corrupt screen memory in display **MODE**s 0, 1, or 2.

It is not necessary to insert a space before any of the parameters following ***FORM**.

Pressing anything other than **Y** in answer to the **GO (Y/N)** prompt results in the formatting sequence being abandoned. Using the ***ENABLE** command immediately before the ***FORM** command results in the **GO (Y/N)** prompt being omitted from the sequence.

If formatting a track fails at the first attempt, a **?** is displayed after the track number concerned. If the track fails to format after six attempts, formatting is abandoned and a disc fault reported.

The minimum abbreviation for ***FORM** is ***FO.** .

*FREE (<drive>)

Purpose

To display, for the specified drive, the number of files that can be added to the catalogue together with the amount of space used and left to be used in terms of sectors (in hexadecimal) and bytes (in decimal).

Examples

```
*FREE

31 Files 31E Sectors 204,288 Bytes Free
00 Files 002 Sectors     512 Bytes Used
```

Displays free space in the current drive.

```
*FREE 1

00 Files 038 Sectors  14,336 Bytes Free
31 Files 2E8 Sectors 190,464 Bytes Used
```

Displays free space in drive 1.

Associated commands

*COMPACT
*MAP

Notes

The minimum abbreviation for *FREE is *FR.

*MAP (<drive>)

Purpose

To display a map of the free space available on the disc. The format is a list of pairs of numbers of the form:

<Sector address> : <length in sectors>

Examples

```
*MAP
Address :  Length
010     :  004
10A     :  023
180     :  050
```

Displays the free space map on the current drive.

```
*MAP 1
Address :  Length
045     :  001
119     :  080
200     :  03A
```

Displays the free space map on drive 1.

Associated commands

*COMPACT
*FREE
*EX

Notes

If there are a large number of entries in the free space map then the disc contents have become fragmented and compaction is needed. Use the *COMPACT command.

The minimum abbreviation for *MAP is *MA.

*ROMS (<rom id>)

Purpose

Produces a list of the installed ROMs. The information displayed consists of the ROM socket number (rom id, see 'Notes' below), the ROM type, the title of the software in the ROM and its version number, if appropriate. If a <rom id> is included, then information is displayed just for the specified ROM (if it is present). The ROM can be identified by its socket number or by its title. The ROM type is displayed as L for a language ROM, S for a service ROM (eg a filing system ROM), or SL for a service/language ROM.

Examples

(Model B Plus only)

```
*ROMS
Rom 15 : ( L) BASIC
Rom 11 : (S ) DFS 2.0
```

Catalogues the installed ROMs.

```
*ROMS 11
Rom 11 : (S ) DFS 2.0
```

Catalogues the ROM with ID number 11.

```
*ROMS BASIC
Rom 15 : ( L) BASIC
```

Catalogues the BASIC ROM.

Notes

The ROM socket numbers for the BBC Microcomputer Model B Plus are detailed under the *FX142 command heading in chapter 43 of the *BBC Microcomputer System User Guide*.

*VERIFY (<drive>) (<drive>) (<drive>) (<drive>).....etc

Purpose

Checks the integrity of each sector of a disc. *FORM<40/80> performs an automatic verification of each track immediately after formatting it.

Examples

```
*VERIFY
Verify which drive ? 1
Verifying drive 1 track <track number>
```

Verifies drive 1. <track number> runs from &00 to &27 or &00 to &4F as appropriate.

```
*VERIFY 0
Verifying drive 0 track <track number>
```

Verifies drive 0.

Associated commands

*FORM40
*FORM80

Notes

If the first attempt to verify a track fails, a ? will be displayed with the track number. If the track fails to verify after five more attempts, verification is abandoned, and a disc fault is reported.

The minimum abbreviation for the *VERIFY command is *V.

Changed commands

The following keywords do not behave exactly as described in chapter 6 of the user guide:

*BACKUP, *DESTROY, *ENABLE

*BACKUP and *DESTROY employ a similar prompt system to the *FORM<40/80> command (see above). They do not need to be enabled, but prior use of the *ENABLE command will result in the GO (Y/N) ? prompt being omitted.

*DRIVE <drive>

The syntax of this command has been changed to

*DRIVE <drive> (40/80)

The new parameter allows an 80 track drive to read 40 track discs by double-stepping the drive head.

Examples

***DRIVE 1 40**

switches drive 1 of an 80-track drive to 40-track disc mode (read only).

***DRIVE 1 80**

sets drive 1 to 80 track operation.

Notes

***DRIVE** <drive> **40** and ***DRIVE** <drive> **80** should not be used with a 40 track disc drive.

Attempting to write using an 80 track drive while in 40 track mode will result in a **Disc read only** error message.

***FORM** ignores the 40 track mode selected with ***DRIVE**. The ***FORM**<**40/80**> command will format the disc size given by the ***FORM**<**40/80**> parameter. Note that an 80 track drive cannot correctly format discs for 40 track drives, and vice versa.

The command ***VERIFY** will check a properly formatted 40 track disc in an 80 track drive set to 40 track operation.

Error messages

The **Drive fault** error message (&C5) has been deleted.

Keyboard links

The interpretation of the keyboard link settings varies slightly from the description given in chapter 12, as shown below:

Link 3	Link 4	Step time
0	0	6 ms
1	0	12 ms
0	1	20 ms
1	1	30 ms (default)

BBC microcomputer 800K drives give optimum performance with a stepping rate of 6 ms. Drive stepping rates can be set with the ***FX255** command. See the *BBC Microcomputer System User Guide* for details.

BBC Microcomputer
Disc Filing System User Guide

Addendum 2

Special Starting Procedure for Protected Discs

Some discs use specially formatted sectors/tracks which are normally reported as 'faulty' by the DFS. This means that ***BACKUP**, ***COPY**, etc. will not read the disc and so the owner of the disc will be unable to reproduce it.

Certain formats which were used in conjunction with an 8271 disc controller IC cannot normally be read by a 1770 disc controller. The special loading program which is on a protected disc will not, therefore, work on a machine using 1770 DFS.

The model B+ is highly compatible with model B machines; the later OS version 2.00 fitted to model B+ does not normally affect its compatibility. However, some 'protected' programs include an additional test which reads the machine Operating System type code, by using **INKEY-256** or its **OSBYTE** equivalent. Unfortunately, this test fails on the model B+ (OS 2.00).

CTRL Z BREAK

A special boot loading mode, **CTRL Z BREAK**, has been introduced into 1770 DFS 2.20 and later. This new mode uses an alternative 8271 emulation technique which may enable otherwise unloadable discs to be booted-in successfuly.

In addition, the special boot mode returns -1 to the **INKEY-256** command and so will improve the compatibility of the model B+ with software written for model B machines.

When to use CTRL Z BREAK

Normally, discs are booted-in by using **SHIFT** and **BREAK** (or **SHIFT D BREAK** if DFS is not your default filing system). This procedure will automatically ***EXEC** a file called **!BOOT** from drive 0.

Always try **SHIFT <D> BREAK** first.

If **SHIFT <D> BREAK** fails to load the disc, try using the special boot mode **SHIFT Z BREAK**, or even **CTRL SHIFT Z BREAK** - (hold down **CTRL**, **SHIFT** and **Z** together, press and release **BREAK** quickly followed by **CTRL**, then release **SHIFT** and **Z** together).

If none of these options will run the disc program, you should check with the software supplier, or the dealer from whom you bought the disc, about its compatibility with 1770 DFS.

September 1985 0403,700 Issue 2 Addendum 2 (0433,051)

The Disc Filing System User Guide

Part no 4Ø37ØØ
Issue no 2
Date July 1983

AUTHOR'S NOTE: Wherever the letters BBC are mentioned in this book, they refer to the British Broadcasting Corporation.

All correspondence should be addressed to:

Technical Enquiries
Acorn Computers Limited
Fulbourn Road
Cherry Hinton
Cambridge CB1 4JN

First published 1983
Published by Acorn Computers Limited
Typeset by Bateman Typesetters, Cambridge

Contents

Introduction

Before you start using the Disc Filing System, check that you have all the following items:

- A single disc drive, or a dual disc drive (two single drives in the same box). Both single and dual drives should be fitted with ribbon cable and power cable.
- A filing system utilities disc.
- A BBC Model B Microcomputer fitted with a disc interface, which should only be fitted by an authorised dealer.

If any of the necessary items are missing then contact your supplier, quoting the order number given to you when you placed your order. The number also appears on the dispatch label.

About this manual

You will notice that some letters, words and phrases in this manual have been printed differently from the rest of the text. This is to help you to tell the difference between explanatory text, words which appear on the computer screen (including BASIC keywords), and certain keys on the computer keyboard.

- Ordinary text appears like this, or *like this* for emphasis.
- Text typed in at the computer or displayed on the screen or BASIC keywords appear `like this`.
- Words like **RETURN** mean that you press the key marked RETURN rather than actually type the letters R E T U R N.

1 What is a disc system?

If you have never used a computer with discs before there are one or two new concepts which you need to learn.

A disc drive

As you probably know, computers have internal memory called Random Access Memory or RAM. When you type in your program it is stored in RAM. However, when you switch off the computer, everything stored in RAM is lost, so if you need the program again, you have to re-type it. To overcome this problem the computer must be able to transfer the contents of RAM into some form of permanent or 'non-volatile' storage before you switch it off. The User Guide which comes with your BBC Microcomputer describes how to use a cassette recorder for this purpose. Transferring a program from RAM to tape is called *saving* it, transferring from tape back to RAM is called *loading* it. The disadvantages of using tape are:

- The process of saving and loading is quite slow.
- You need to keep track of where on the tape each piece of information is, so that you do not record over it.
- You have to wind the tape to the right place yourself.
- Winding from one end of the tape to the other is slow.
- It is not possible to wind the tape to a particular point accurately.

A disc system does not have these disadvantages. To help you to understand how a disc system works, we shall draw some comparisons with a filing cabinet. A disc system always includes at least one disc drive. The BBC Microcomputer's disc drives are buff-coloured metal boxes approximately 15.5cm × 9cm × 24cm. The front of a disc drive is black. On the front is the BBC owl, a small red light and a spring flap called the disc drive door. The door can be opened or closed, but must be closed while the disc drive is supposed to be working. These are floppy disc drives, as distinct from fixed disc drives, which you may have heard of. The disc drive can be compared to an empty filing cabinet with no drawers in it yet.

Just as a filing cabinet is pretty useless without drawers, so a disc drive cannot do much without discs. The discs used with the BBC Microcom-

puter are 5¼" soft-sectored floppy discs. They can be inserted and removed from the disc drive via the slot in the front. There is one right way of inserting them, and several wrong ways. As in the case of the filing cabinet drawers, putting them in the wrong way is a waste of time. Fig 1 shows the correct way to insert a disc. To stress this:

Discs should be inserted with the label upwards and with the edge nearest the label in your hand. The edge opposite to the label and the read/write slot of the disc go in first.

Fig 2 shows and names the various parts of a disc and chapter 3 gives more detailed information about them. The discs hold the information. Just as you can put different cassettes in a cassette recorder, you can also put different discs into a disc drive; but the computer can only read information from one disc at a time. A disc may have lots of different groups of information on it, and these groups are called 'files'. Files can have any information in them. Typical examples would be one of your programs or some data generated by a program which you wish to keep.

Returning again to the comparison with a filing cabinet, opening a drawer and throwing all the papers into it at random would make it difficult to find them again. To solve this problem, people usually put dividers into a filing cabinet drawer, and often these are labelled alphabetically. The result is that the information is grouped so that you can find it again quickly. The same principle is followed when the computer puts information on to a disc. When you first buy discs, they are blank – like empty drawers. Before the computer can put any information on them, the discs must first be prepared by having marks put on them, which divide the discs into sectors. A 'sector' is the name given to a set of equal divisions created on the disc by the computer. (See Fig 5.) This operation is called 'formatting' and is fully described in chapter 7.

What the disc drive does

When you insert the disc into the disc drive and close the drive door, a rotating boss engages with the central hole in the disc and spins the magnetic disc inside its protective jacket. (Do not confuse the protective jacket with the disc envelope – see Fig 2.) In order to read or write information on to the disc the disc drive has a 'read/write head'. This head is designed to move in and out along the 'head slot' in the disc jacket. This head actually rests on the surface of the magnetic disc as it rotates inside

the jacket. When you want to read some of the information on the disc, you give the computer the name of the file containing that information. The computer will move the read/write head to the sector on the disc where the start of the information in the named file is recorded. This is equivalent to you opening the filing cabinet drawer, looking along the dividers until you find the one you want and then preparing to remove the relevant file for reading.

At this point it is worth noting that your files may be too large to fit into the fixed size of one sector. This is no problem. A file always begins in a new sector but may occupy a number of sectors following the first. Each sector can hold up to 256 characters or 'bytes'.

Disc Filing System

The main disadvantage of using a cassette recorder to store information is that you have to control the cassette recorder and keep track of the information on it.

When using a disc this is all done for you by the 'Disc Filing System'. The Disc Filing System is a machine-code program produced by the computer manufacturer. On the BBC Microcomputer it is stored in a special kind of memory inside the computer called 'Read Only Memory' or ROM. The program is not lost when you switch the computer off. Once installed, it is always there. All the actions of the disc drive are controlled by the computer using this program. When you prepare new discs by formatting them this is done by the Disc Filing System. When you `SAVE` one of your BASIC programs the Disc Filing System does the following:

- Starts the disc drive working.
- Finds a free place on the disc big enough for your program.
- Makes a note of where it put your program so as to be able to find it again.
- Moves the disc drive's read/write head accurately to the start of the first sector in the free space.
- Transfers a copy of your program from the RAM to the disc.
- Stops the disc drive.

All this is done without you having to think about it and is quite a bit quicker than saving a program on to a cassette tape.

When you save a program you have to give it a name. This is true for the disc system as well as the cassette system. However, the Disc Filing System puts the name to special use. The first two sectors on every disc are reserved for a 'catalogue' when the disc is formatted.

The name of your program, referred to as a file name, is written into the catalogue together with the number of the sector on the disc where the information starts. (Note it may continue over several sectors.) When you want the file containing your program back again you simply type `LOAD "file name"`. The filing system checks the catalogue to find out where on the disc to find the file, and then moves the read/write head to that exact place on the disc. The file is then loaded into the computer's memory (RAM) automatically. This illustrates another advantage of a disc drive. The read/write head can be quickly moved to any point on the disc with great accuracy. (Incidentally, the precision engineering needed to accomplish this explains why disc drives cost so much more than cassette recorders.) Because of this accuracy, a number of other facilities are available besides loading and saving programs. These include the ability to copy, delete, build and rename files. Additional facilities let you examine a disc catalogue, restrict access to files or move directly to specific points within a file.

As a final comparison, imagine an automatic filing cabinet where to find something all you have to do is specify the name of a document and paragraph number within it. The filing cabinet drawer opens, the correct divider is selected, the document is located and then presented to you open at the appropriate page. You may see why microcomputers have become so popular in offices.

Controlling the filing system

The filing system controls the disc drive, and we must be able to give instructions to the filing system. Two ways are provided. One is by typing a '`*`' character, followed by a special command. These are all listed in chapter 6 together with details of their functions. Any of these direct commands can be incorporated in a program if required. Chapter 8 describes the use of a number of BASIC keywords, with special reference to files created on a disc. All these keywords are introduced in the *BBC Microcomputer User Guide*.

Summary

A disc system includes either one or two disc drives, some discs, a connection to the computer and a machine-code program permanently in the computer called a 'Disc Filing System'.

A disc is inserted into a disc drive where it spins round inside its protective jacket.

The disc drive's read/write head moves in and out along a radius of the disc as it spins around.

The Disc Filing System controls the disc drive and the movement of the head.

Discs are divided into sectors by the filing system and the first two sectors are reserved for a catalogue.

Programs and other information are stored on the disc and are given a 'file name'.

By referring to the catalogue, the filing system can find any information on a disc associated with a specified file name.

Files can occupy more than one sector.

Procedures are provided to locate particular points in files.

Instructions may be given to the filing system by direct command or from within a program.

2 Getting going

With the power turned off, connect the two cables from the disc drive to the underside of the computer as shown in Fig 1. The plugs are designed so that they will only fit one way. The plug on the power cable is shaped like a rectangle with two adjacent corners cut off. *Do not* force it in the wrong way round. Sometimes the plug on the ribbon cable has a lump on one side which locates into a notch in the socket on the computer. Alternatively, you may have to try both possible ways before you get it right. When the drive is connected, turn on the power and press **BREAK**. The following message, or one very similar, should appear on the screen:

```
BBC Computer

Acorn DFS

BASIC

>
```

This indicates that the Disc Filing System is installed and working OK. Now press the **SHIFT** key and tap the **BREAK** key, then release the **SHIFT** key. The disc drive motor will start turning for a couple of seconds and the red light on the front of the drive will come on. This shows that the disc drive is properly connected and working. If nothing happens, check the connections between the disc drive and computer.

The foregoing assumes that the auto-start option has been set to work when **SHIFT** is held down with **BREAK**. (See chapter 4.)

Insert the utilities disc into drive Ø. (With dual drives, drive Ø is the one at the top. See Fig 1.)

Press **SHIFT** and **BREAK** again. This time the following message will appear on the screen:

```
Hello, this is the BBC microcomputer.
This message was automatically loaded.
```

The utilities disc contains three useful programs:

*FORM40 formats a 40 track disc
*FORM80 formats an 80 track disc
*VERIFY verifies a disc

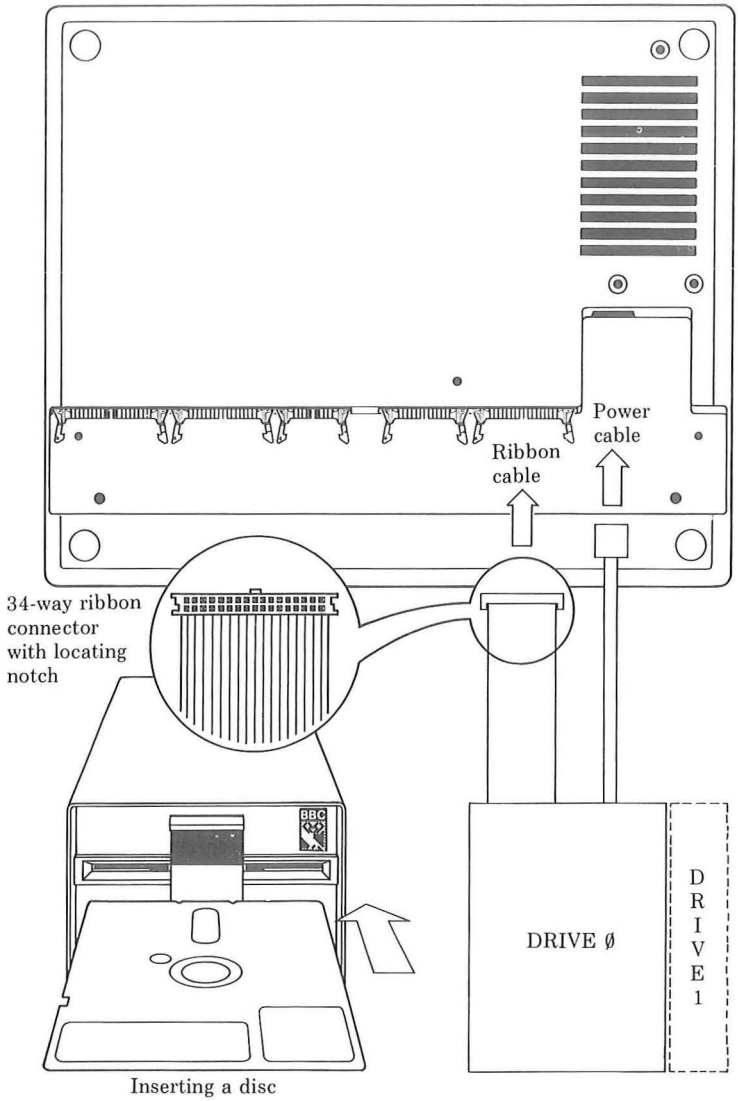
These programs are used by typing their name.

There are also a number of demonstration programs on this disc.

Press the SPACE BAR to see them run.

This is an example of a !BOOT file which will be explained in more detail later. Now press the SPACE BAR to continue. This will run the series of demonstration programs supplied on the disc. When the > prompt reappears on the screen you can start entering your programs or filing system commands, but before you do that, read the next two chapters, 'Discs' and 'Disc files'.

Fig 1 Connecting the disc drive



3 Discs

The BBC Microcomputer uses 5¼" soft-sectored floppy discs for storing information. You may have heard them referred to as 'floppy discs', 'discettes', or 'mini-discs'; we will always refer to them simply as discs. (The American spelling is disk.)

Two types of disc drive are available for the BBC Microcomputer, a single unit (1ØØKbytes) and a dual unit (8ØØKbytes). Discs used on one type will not work on the other. A disc from a single drive will store about 1ØØ,ØØØ characters, equivalent to 5Ø pages of this book. Discs from a dual drive unit will store about 4ØØ,ØØØ characters each. It is possible to use two single disc drives linked together providing space for storing about 2ØØ,ØØØ characters, but this is not a standard option.

Handling

Discs should be handled with care to avoid physical damage or damage to the recorded information. Fig 2 will help you to identify the various parts of the disc that we are referring to. The following guidelines should be observed:

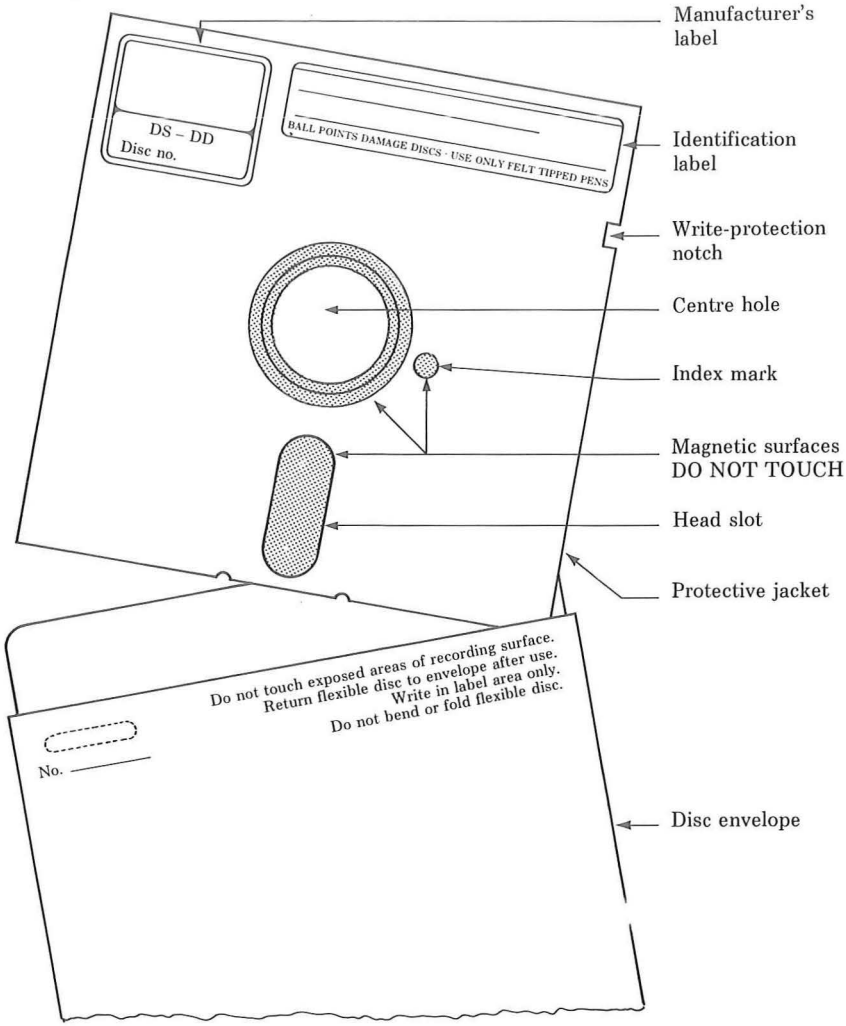
- Do not try to remove the circular magnetic disc from the square, black protective jacket covering it.
- Do not touch the exposed recording surfaces.
- Avoid dust. Put the discs back into their envelopes when they are not in the disc drive.
- Do not bend them, drop them on the floor or put heavy objects on them.
- Keep them in a storage box designed for the purpose.
- Keep them away from strong magnetic fields such as those generated by televisions, monitors, tape recorders, telephones, transformers, calculators, etc.
- Avoid excessive heat, moisture and direct sunlight.
- Only use felt-tipped pens to write on the labels and don't press hard.
- Insert discs into the drive carefully. If it rotates noisily open the drive door and adjust it. If you get an error message such as `Disk fault` or `Drive fault`, then the disc probably hasn't centred in the drive. To overcome this, make sure that the drive spindles are

rotating by addressing the drive before closing the drive door. This is especially important when a new disc is inserted.

Information is packed quite densely onto the disc, so it is sensitive to even very small scratches and particles of food, dust, tobacco, etc.

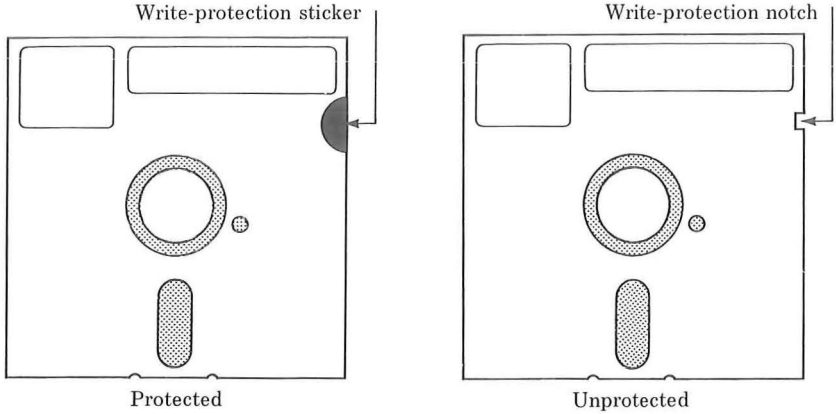
The foregoing is deliberately comprehensive but do not let it frighten you from using the discs. Handled sensibly, a disc will give good service.

Fig 2 A 5¼" disc



Preventing accidental erasure

You will notice from Fig 2 that there is a small notch in the side of a disc called the 'write-protection' notch. It is used to protect the information on a disc from being overwritten. Every box of discs is supplied with a number of adhesive tabs which can be used to cover the write-protection notch in the disc. The diagram below shows that covering the notch with one of the adhesive tabs will prevent the disc drive from writing to the disc and from deleting anything on it. Reading the existing information on the disc is still allowed.



Copying information

Another way of protecting important information is to keep several copies of it on different discs. Where computers are used in business, industry or other activities which use large volumes of information, a standard routine for this has been evolved. It is often called the 'grandfather, father, son' principle of copying information, and is good practice for anyone using a disc storage system. It works as follows:

Day 1, MASTER copied to GRANDFATHER
Day 2, MASTER copied to FATHER
Day 3, MASTER copied to SON

As you can see, it involves keeping three separate discs, each with a copy of the information from the master disc on it. On day 4 the master would be copied to the grandfather again and so the cycle continues. In business, where information on disc changes from day to day, this regular

routine is important. For personal computing it may not be so vital, but you will want to keep several copies of important programs and information which you have worked hard to produce. The filing system provides two facilities which make copying information easy. These are *BACKUP and *COPY which allow you to copy a complete disc, or specified sections of it. See Figs 3 and 4 and the appropriate sections of chapter 6 for full details.

We suggest that you make a copy of your utilities disc now!

Proceed as follows

If it has not been done already write-protect the utilities disc with a tab, as described. This is very important otherwise you might lose all the information on the disc.

Insert the utilities disc into drive Ø and type

```
*FORM 4Ø RETURN
```

or (if you have a dual drive 8Ø track unit)

```
*FORM 8Ø RETURN
```

Remove the utilities disc and put a new disc in its place. Do not forget to change the disc over. If you format the utilities disc you will lose everything on it.

In reply to the question displayed, type

```
Y
```

What the computer now does is called 'formatting' which prepares the new disc for use with the BBC Microcomputer. A series of numbers are displayed, and when the formatting is finished the word Formatted appears. Now remove the new disc and put the utilities disc back again.

If you have a dual drive, put the new disc into drive 1 (the other drive) and type

```
*ENABLE RETURN
*BACKUP Ø 1 RETURN
```

The computer will now make a complete copy of the utilities disc on the new disc.

If you have a single drive, type

```
*ENABLE RETURN
*BACKUP 0 0 RETURN
```

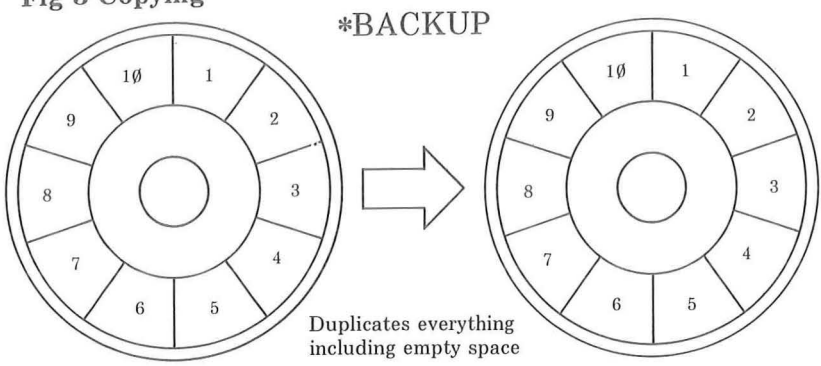
then follow the instructions on the screen to insert the utilities (master) disc and the new disc alternately. (See Fig 4.)

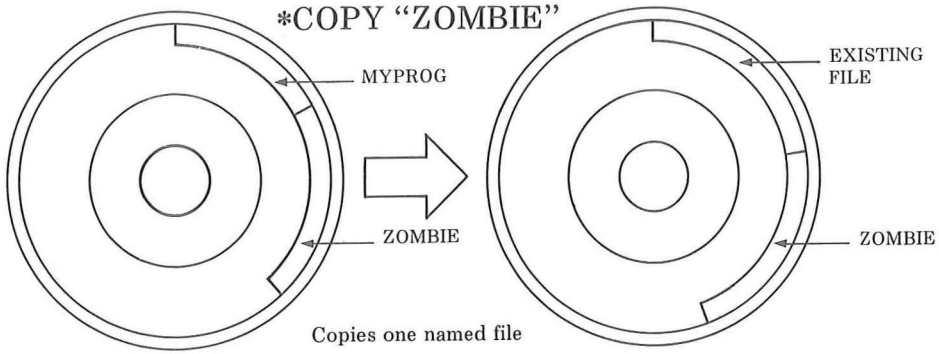
If the message

```
Disk read only
```

appears, it means you had the wrong disc in the drive. Start again from ENABLE.

Fig 3 Copying





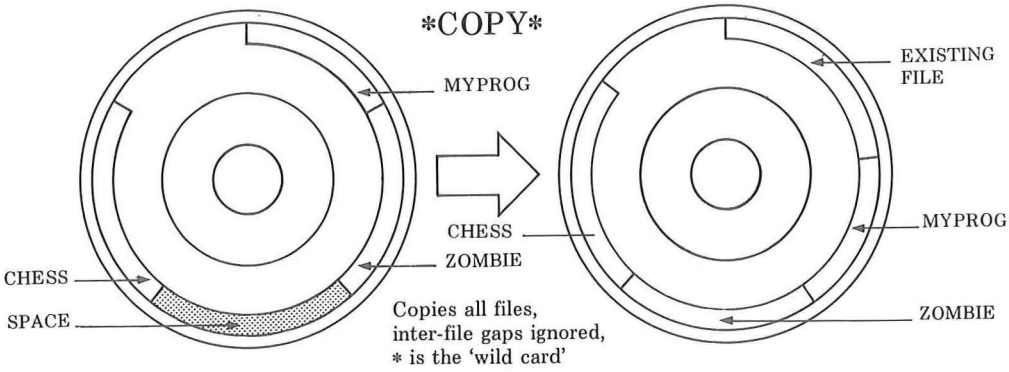
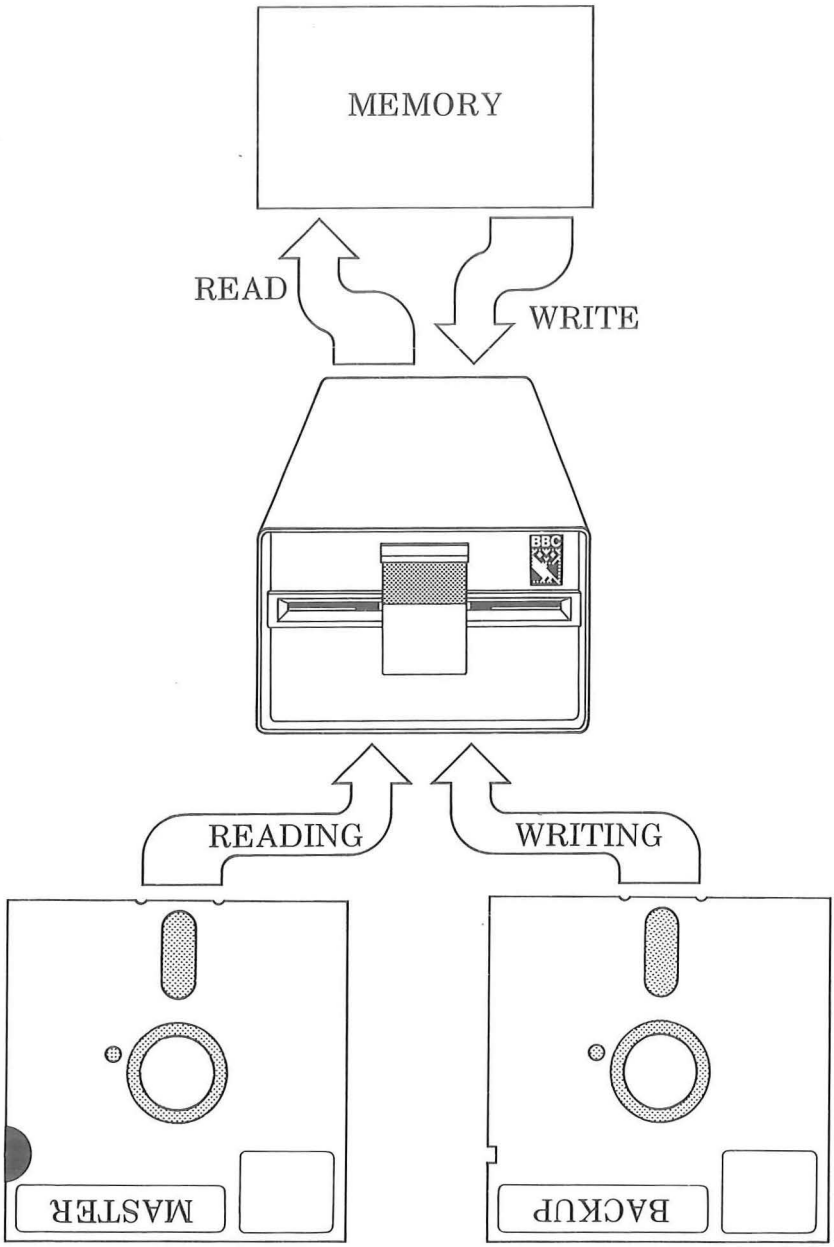


Fig 4 Single disc drive copying



Tracks, sectors and bytes

Information is written on to the disc in concentric circles, called tracks. Each track is divided into ten sectors. Each sector is further divided into 256 bytes. Space on the disc and in the computer's memory is measured in bytes, and one byte corresponds to one character. 256 of the bytes in each sector are available for storing your programs and data. From this it follows that a 40 track single-sided, single track density disc of the type used in the single drive unit will hold:

40 tracks × 10 sectors × 256 bytes = 102400 bytes

(or characters) of information. The dual drive unit uses double-sided, double track density discs. Therefore with 80 tracks on each side, one disc will hold:

160 tracks × 10 sectors × 256 bytes = 409600 bytes

(or characters) of information.

What is formatting?

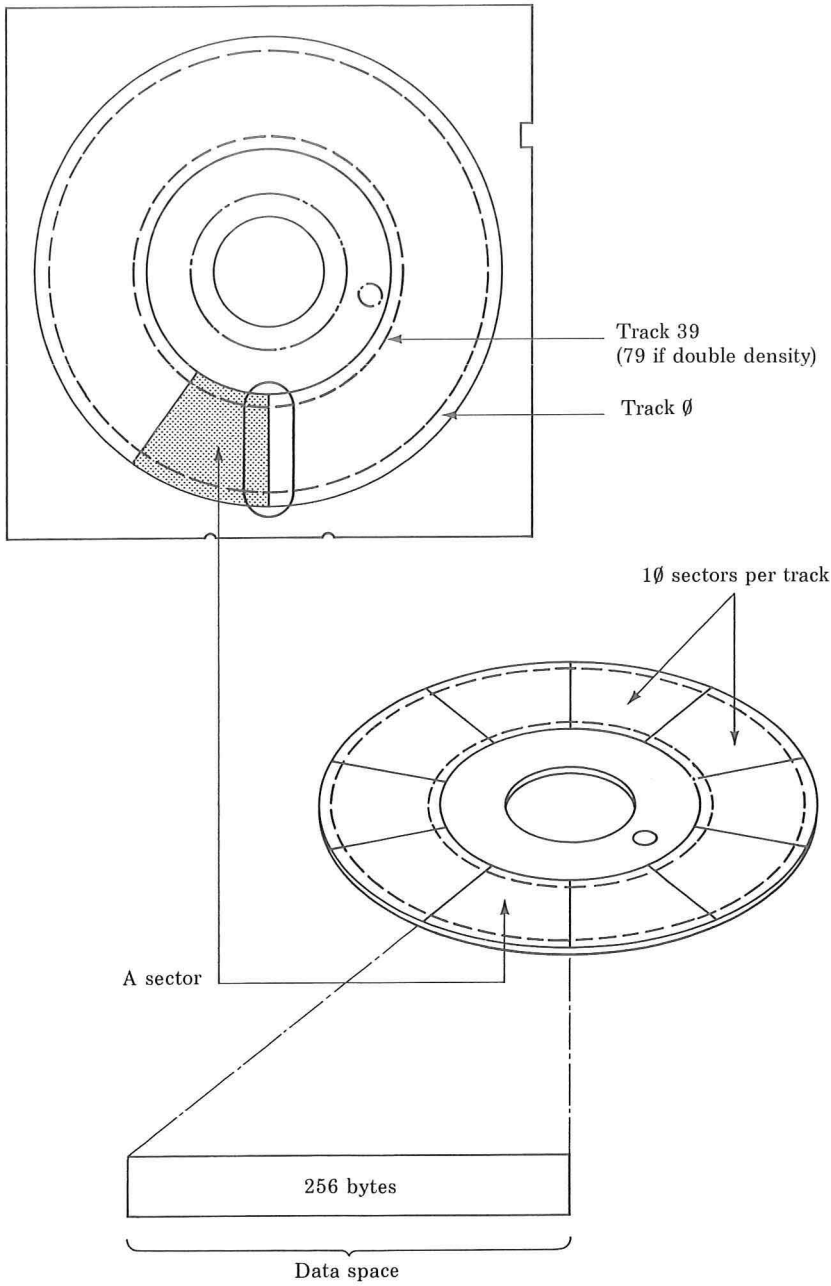
The Disc Filing System automatically records the location of your programs and data on the disc. The first two sectors on track zero of a disc are reserved for this purpose. The 'catalogue' of a disc is recorded in these sectors. Whenever you wish to access a piece of information on the disc, the filing system reads the catalogue first to find out where on the disc it can be found. The tracks and sectors are the reference marks on disc which make this possible.

Clearly before a disc can be used by the filing system for storing your information it must have these 'reference marks' put onto it. All new discs must be prepared in this way. The process is called 'formatting'. It includes setting up the track and sector format on the disc and creating the catalogue. (Full details are provided in the technical information in chapter 12.)

The filing system utilities disc has two programs on it called `*FORM 40` and `*FORM 80` which perform this task for 40 and 80 track discs. Each sector on each track is given a unique three-digit identifier in the catalogue. The first two digits are the track number, the last one is the sector number. The sectors are numbered 0 to 9 and the tracks 00 to 39 (or 79 if it is an 80 track disc).

The important thing to remember is that new discs must be formatted before you can use them with your computer.

Fig 5 Tracks, sectors and bytes



4 Disc files

Probably the first thing you will want to do with the filing system is to record one of your programs onto a disc. You can do this simply by using the SAVE command in BASIC, and the filing system takes care of the rest. (*Note:* Not to be confused with *SAVE described later in this manual.) When you have typed the program into the computer the SAVE command causes it to be copied onto the disc. When SAVEd, the program must be given a name. This is called the file name and is used to identify the program when you want to copy it back from the disc into the computer's memory. Each program SAVEd onto the same drive must be given a unique name. The format of the SAVE command is

```
SAVE  "filename"
```

where "filename" can be up to seven characters. Letters and digits are allowed. The characters

```
#     *     .     :
```

have special meanings which are explained later.

The file name is written into the catalogue together with the sector number where the information starts. Next time you refer to the file name, the filing system checks the catalogue to see where the information has been placed on the disc, and the old file is deleted and replaced by the new one. The filing system ensures that each new file begins with a new sector.

File specifications

The full specification for a file is

```
:Drive number.Directory.File name
:<drv>.<dir>.<filename>
```

eg:

```
:1.Z.MYPROG1
```

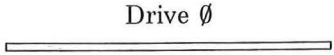
Notice the drive number, directory and file name are separated by full-stops. These are needed so that the computer can distinguish the separate parts of the file specification.

Drive numbers

Drive numbers must be in the range Ø to 3 and preceded by a : (colon). The colon tells the computer: 'This is the start of a file specification, the drive number follows.'

The drives are numbered as shown below. Notice that each side of a double-sided disc is given a separate drive number.

Single drive, single-sided



Dual drive, double-sided

The effect of including the drive number in the full specification is that

`:1.$.MYPROG 1` is different from `:2.$.MYPROG 1`

Although the file names are the same, they are on different drives.

Directories

The directory is a single character used to divide the catalogue into independent sections. Files of the same name can be created on the same disc with different directories. Although on the same drive,

`:1.$.MYPROG` is a different file from `:1.A.MYPROG`

because the directory is different.

File names

The file name can be up to seven of most of the characters on the keyboard in any combination, except # * : . as we mentioned earlier. When we need to refer to the complete file specification in future we will use the abbreviation <fsp>.

When the filing system is started by pressing BREAK or SHIFT BREAK, the current directory and drive number is always set to drive 0 and directory $. The drive and directory can therefore be omitted from file specifications. They will be assumed to have these values.

Typing

```
SAVE "MYPROG"
```

will automatically store your program in a file named

```
:0.$.MYPROG
```

assuming you have not changed the current drive and directory. (Chapter 6: 'The filing system commands' explains how you can change the current drive and directory with the commands *DRIVE and *DIR.)

Multi-file operations

Another common term used to refer to multi-file operations is 'wild card' facilities. Some of the filing system commands can operate on a number of files instead of just one. These are all followed by the abbreviation <afsp> instead of <fsp> (<afsp> stands for 'ambiguous file specification'). *INFO is an example of such a command. It provides information about a named file, eg:

```
*INFO :0.$.MYPROG          RETURN
```

will display information about the file named MYPROG in directory $ on drive 0.

However, it is possible that you want information about a number of files. The 'wild card' facilities enable you to specify several files for the command to operate on. The wild cards are provided by the characters * and

which have special meanings when they appear in the file specification, eg:

```
*INFO  :0.#.MYPROG
```

means: 'Display information about files called MYPROG in any directory on drive 0'.

```
*INFO  :0.$.MYPRO#
```

means: 'Display information about all files on drive 0 in directory $ with names starting "MYPRO" followed by any single character.' eg:

MYPROA, MYPROT and MYPROG and so on.

```
*INFO  ###        RETURN
```

gives information on all files with three-letter file names in the current directory.

The character * means multiple #s to the end of the field, eg

```
*INFO  :0.$.M*
```

will display information about any files on drive 0 and directory $ whose names begin with M.

```
*INFO  *A*        RETURN
```

gives information on all files in the current directory with an A in them: for example, A, AB, FREDA, PGRAM1 etc.

```
*INFO  *.*        RETURN
```

gives information on all files, in all directories.

Auto-start facilities

Sometimes it is useful to make a program or a file on one of your discs *LOAD, *RUN or *EXEC automatically when you insert the disc and press SHIFT and BREAK. This can be done using a file named !BOOT. !BOOT is a special file name recognised by the filing system when you start the computer by pressing SHIFT BREAK. If there is a file of specification

```
:0.$.  !BOOT
```

the filing system will do one of four things according to the option set on the disc using *OPT 4,n (see chapter 6).

Option 0: ignores !BOOT
Option 1: *LOADs !BOOT into memory
Option 2: *RUNs !BOOT as a machine-code program not a BASIC program
Option 3: *EXECs !BOOT

See chapter 6: 'The filing system commands' under the section *EXEC for an explanation of option 3. That section also describes how to use this auto-start facility to make the computer run one of your BASIC programs automatically.

The options can be changed using the *OPT 4 command. The 'Hello' program on the filing system utilities disc is loaded using a !BOOT file.

As well as programs, you may wish to store data on the discs. The filing system provides facilities for storing and retrieving the data quickly and selectively under the control of your programs.

One of the methods is to use a type of file called a 'random access file' – see chapter 8.

Library directory

The Disc Filing System enables you to specify one drive/directory as the 'library'. This will always be set to :0.$ when you start the computer by pressing BREAK. It can be altered using the filing system command

*LIB, until the next BREAK. All the utility programs should be located in the library. This is because when you type

* (utility name) RETURN

it is equivalent to typing

*RUN (utility name) RETURN

where the drive and directory are omitted and will be assumed to be either the current drive/directory or the library. The filing system will first search the current drive/directory for the file and then, if it cannot find it there, it searches the library.

5 The BREAK key

The BREAK key resets the BBC Microcomputer. However, the Disc Filing System can preserve some of its status after BREAK. There are two types of BREAK: the first is called 'hard break', and is achieved by holding down the CTRL key and pressing BREAK; The second is called 'soft break', and is done by just pressing the BREAK key.

As far as the Disc Filing System is concerned, hard break is the same as switching the computer off and then on again. The directory and library are set to :0.$ and any open files are forgotten about. Data written to a file which is still open may be lost. (Cold start.)

When you do a soft break, the Disc Filing System preserves its status, ie the directory and library remain the same, and open files remain open, but again – data may be lost. (Warm start.)

If you press SHIFT BREAK to make the Disc Filing System auto-start using the !BOOT file, the Disc Filing System does a cold start as per CTRL BREAK.

Important: To do a warm start, as per a soft break, or going *TAPE then *DISC to restart the DFS, the DFS must assume that the contents of RAM below the default value of PAGE have not been corrupted. This means that if you have been using proprietary software which uses all the RAM from &E00 upwards, you must exit the program with a hard break to obtain normal DFS operation.

6 The filing system commands

The Disc Filing System is an 8K byte program. BASIC programs are stored on a disc or tape, but the filing system is stored in Read Only Memory (ROM) inside the BBC Microcomputer. The filing system controls the reading and writing of information to and from the discs and provides a number of useful facilities for maintaining that information. The following pages describe all the filing system commands. They are words which the filing system program will recognise and act on. They can be typed directly on to the keyboard or embedded within your BASIC program. They are all prefixed with the * character which signals the computer that a filing system command follows. Each command is described under a number of sections with headings as follows:

Command
This is followed by a syntax abbreviation and a few words explaining the derivation of the word.

<drv> = drive
<fsp> = file specification
<dir> = directory
<afsp> = ambiguous file specification: this means that you can get the computer to perform an operation on a number of files at a time by typing in as much of the file names as is common to all these files – followed by a * character. For example, if you want to `COPY` the files called `FRED1`, `FRED2` and `FRED3` from drive 1 to 2, you could type

```
*COPY 0 2 FRED*        RETURN
```

Purpose
A plain English description of what the command does.

Example
This section gives a few one-line examples of the use of the commands. These examples are only intended to be illustrative.

Description
A description of the command using normal computer jargon.

Associated commands
This section lists commands which have similar functions or are normally used in conjunction with this command.

Demonstration program
Where it is needed, a short program is included to illustrate use of the filing system command in a BASIC program.

Notes
Particular points to watch for or special applications of the command are covered by additional notes if necessary.

Diagrams
Diagrams are used where they make the function of a command clearer.

*ACCESS <afsp> (L)

Purpose
To prevent a file from being deleted or overwritten. The command 'locks' or 'unlocks' a file. You cannot delete, overwrite or write to a locked file until you unlock it again. If you load a file which is locked, you will not be able to save it again with the same name. This is because saving a file with the same name as one already on the disc causes the one on the disc to be deleted and replaced with the new file. A locked file cannot be deleted.

Example

```
*ACCESS HELLO L
```

This locks the file HELLO.

```
*ACCESS HELLO
```

unlocks it again so that it can be deleted or overwritten.

Description
Sets or removes file protection on a named file. It prevents a number of other filing system commands from acting on the file.

Notes
Once locked, a file will not be affected by the following commands:

```
*SAVE
*DELETE
*WIPE
*RENAME
*DESTROY
```

If you attempt to use any of these commands on a locked file the message

```
Locked
```

is produced.

If you attempt to use *ACCESS on a write-protected disc the message

```
Disk read only
```

is produced.

Important: Locking a file does *not* prevent if from being removed from a disc with *FORM40 or *FORM80 or from being overwritten with *BACKUP.

*BACKUP (source drv) (dest.drv)

Purpose
To read all the information on one disc and write it to another, producing two discs with identical information.

Example

```
*ENABLE
*BACKUP  Ø  1
```

copies all the information on drive Ø onto drive 1.

Description
Sector by sector copy program.

Associated commands

```
*COPY
*ENABLE
```

Notes
*ENABLE must be typed before the command will work, otherwise the message

```
Not enabled
```

is displayed.

If you give Ø as the source and destination drives, eg:

```
*BACKUP  Ø  Ø
```

the program will alternately ask you to insert the source and destination discs into drive Ø. This makes it possible to copy discs even if you only have a single drive. A 4Ø track disc is copied in five sections.

Fig 4 in chapter 3 illustrates the process.

All information previously on the destination disc is overwritten, so be careful not to confuse the source and destination discs. If the source disc is blank, the destination disc will end up blank as well.

Warning: The contents of memory may be overwritten by this command. If you have a program or some data in memory that you want to keep, SAVE it before you use the command.

*BUILD <fsp>

Purpose
To create a file directly from the keyboard. After typing this command everything else entered will go into the named file. This is useful for creating EXEC files and the !BOOT file described in chapter 4.

Example

```
*BUILD  !BOOT
```

will cause everything subsequently typed in to be entered into a file called !BOOT.

Line numbers are displayed on the screen to prompt you to enter your text as follows:

```
>*BUILD  !BOOT
0001  FIRST  LINE  OF  TEXT
0002  SECOND  LINE
0003  ESCAPE
```

Pressing the ESCAPE key on a line by itself terminates a *BUILD command.

Description
Builds a file from the keyboard.

Associated commands

*EXEC
*LIST
*TYPE

*CAT (<drv>) catalogue

Purpose

The command displays the catalogue of a disc on the screen, showing all the files present on the disc. (drv) is the number of the drive you want displayed. If (drv) is omitted, the current drive is assumed.

Example

```
*CAT 0          RETURN
PROGRAM (nn)
Drive: 0                    Option: 2 (RUN)
Directory: 0.$               Library: 0.$

     !BOOT                       HELLO
     SUMS                        TABLE
     TEST                        VECTORS
     ZOMBIE

A. HELLO L                  B.SUMS
```

Note that the heading part of the catalogue shows the drive number, the title of the disc, the currently set auto-start option of the disc (in this case 2 for RUN), and the currently selected library and directory. The files are displayed in alphabetical order reading across the two columns. In the example above there are nine files on the disc.

!BOOT to ZOMBIE are in the current directory $. The current directory's files are always listed first. A. HELLO is in directory A. It is also followed by L, meaning that it is a 'locked' file. (See *ACCESS for an explanation.) B. SUMS is in directory B and is not locked.

Description
Displays a disc catalogue.

Associated commands

```
*INFO
*ACCESS
*TITLE
*OPT 4, n
*DIR
*DRIVE
```

*COMPACT <drv>

Purpose
Attempting to SAVE a program or file on to a disc may produce the message Disk full if there is no single space available on the disc big enough for the information. It may be that there is enough space, but it is split into several small sections. This command appends all spare space on a disc to the end. When you delete a number of files, the spaces they had occupied will probably be distributed over the disc with current files in between them. *COMPACT moves all current files to the 'start' of the disc leaving the space in one continuous block at the end.

Example

```
*COMPACT 1
$.HELLO  1700  801F  0003B  002
$.SUMS   1700  801F  00098  003
```

. . . and so on.

As 'compacting' proceeds, all the current files are displayed in the order in which they occur on the disc.

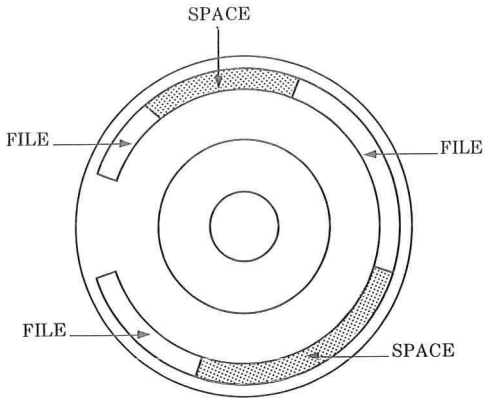
Description
Moves all available space on a disc into one continuous block following the current files.

Associated commands

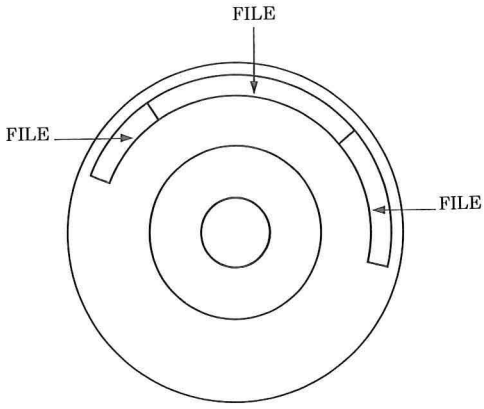
*SAVE and BASIC's SAVE and OPENIN.

Notes
This facility will only do anything if there is space between the files. There will only be such space if a file has been deleted from between two others. Compacting a disc happens faster if you select mode 7 beforehand.

Fig 6 *COMPACT



Before



After

Warning: This command may overwrite the contents of memory. If you have a program or data in memory that you want to keep, save it before you use this command.

*COPY <source drv> <dest. drv> <afsp>

Purpose
To copy a named file or files from one disc to another.

Example

```
*COPY 0 1 HELLO
```

This copies a file called HELLO in the current directory on drive 0 onto drive 1.

Description
File copy program.

Associated commands

```
*BACKUP
```

Notes
The 'wild card' facilities may be used to specify a group of files to be copied, eg:

```
*COPY 0 1 #.MY*
```

Copies all files beginning MY irrespective of which directory they are in. Information already on the destination disc is not affected.

Diagram
Figs 3 and 4 in chapter 3 apply.

Warning: This command may overwrite the contents of memory. If you have a program or data in memory that you want to keep, save it before you use this command.

*DELETE <fsp>

Purpose
To remove a single named file from the catalogue of a disc. The space occupied by the file becomes available for other information. Succeeding file names in the catalogue are shuffled up, but not the files themselves. Once a file is deleted you cannot get it back again.

Example

```
*DELETE FRED
```

removes a file called `FRED` from the current directory on the current disc.

Description
Single file deletion.

Associated commands

```
*WIPE
*DESTROY
*COMPACT
```

Notes
If the disc is write-protected the message

```
Disk read only
```

is produced. If the file is not found in the directory, the message

```
Not found
```

is displayed. If the file is locked, the message

```
Locked
```

appears.

Once deleted, a file cannot be restored.

*DESTROY <afsp>

Purpose
To remove specified files from the disc in a single action. This command may take the ambiguous file specification so that groups of files can be deleted. When you use this command, a list of the files to be deleted is displayed. A single Yes/No question appears at the end of the list offering you the choice to go ahead and delete all the listed files or not. Use this command with care, because its effect is not reversible. It will not attempt to remove locked files. (See *ACCESS.)

Example

```
*ENABLE
*DESTROY  *.H*
A.HELLO
$.HELLO

Delete  (Y/N)
```

If you type Y in reply to the question all the named files will be deleted. The message

```
Deleted
```

is displayed when the job is done. Typing anything else cancels the command.

Associated commands

```
*ENABLE
*WIPE
*DELETE
```

Notes
Once destroyed, files cannot be restored.

*ENABLE must be typed immediately before *DESTROY, or it will not work and the message

```
Not  enabled
```

is displayed.

*DIR (<dir>) Set the directory

Purpose
To change the current directory to (<dir>). After a cold start, the current directory is always set to '$'. To save files in a different directory in the catalogue, you must use this command to change the current directory to the one you want and then SAVE them.

Example

```
*DIR A
```

This sets the current directory to A. You now have access to any files in directory A in the catalogue. Any files now saved using *SAVE, or BASIC's SAVE, will be in directory A.

Description
Sets the current directory to the argument supplied.

Notes
Directory can be set to any character except these four exceptions

```
#     *     .     :
```

This command does not alter the directories written in the catalogue. It merely states which directory in the catalogue you have access to by default.

You can change drive at the same time, and this takes the form

```
*DIR :<drv>.<dir>
```

So if the current drive is drive 0, and the current directory is $, then

```
*DIR :2.A
```

will set the drive to drive 2, and the directory to A. Alternatively, you can set the drive separately using the *DIR command:

```
*DIR :2
```

*DRIVE <drv> Set current drive

Purpose
Changes the current drive to <drv>. Any commands which follow will work on <drv> until another drive is specified.

Example

```
*DRIVE 1
```

sets the current drive to 1 and

```
*CAT
```

will produce a catalogue of drive 1.

```
*CAT 0
```

will catalogue drive 0, but the current drive is still drive 1 until you change it back to 0, or after a cold start.

Description
Sets the current drive, and leaves the current directory unchanged.

*DUMP <fsp>

Purpose
Produces a hexadecimal listing of a file on the screen.

Example

```
*DUMP  SUMS
```

Description
Hexadecimal screen dump.

Associated commands

*LIST
*TYPE

Notes
It is useful to use this command in page mode so that the file is displayed one page at a time on the screen.

CTRL N selects page mode, CTRL O turns it off.

*ENABLE

Purpose
Some of the filing system commands produce irreversible effects. To prevent them from being used accidentally, it is necessary to type *ENABLE before they become operational. These commands are:

```
*BACKUP
*DESTROY
```

Example

```
*BACKUP
```

will not work; the message

```
Not enabled
```

is produced.

```
*ENABLE
*BACKUP
```

will work.

Notes
*ENABLE must be typed immediately before the command to be enabled. Any * name command typed in between nullifies the *ENABLE.

*EXEC <fsp> execute

Purpose
This command reads byte by byte all the information in a named file as if it was being typed on the keyboard. This is useful when you find that you are repeatedly typing the same sequence of commands. Instead you can build an EXEC file consisting of all these commands and type *EXEC <fsp> each time you want this sequence of commands. *BUILD <fsp> is an associated command used to create an EXEC file.

Example

```
*EXEC HELLO
```

takes the contents of file HELLO and reads it one character at a time as if it was being typed at the keyboard.

Description
Executes the contents of a named file by reading each byte as if it were coming from the keyboard.

Associated commands

*BUILD

Notes
One useful application of the *EXEC command is in association with the auto-start facilities described in chapter 4 and in the section on *OPT4 in this chapter. If you create a !BOOT file containing the BASIC keyword CHAIN followed by the file name of one of your BASIC programs, the effect of pressing SHIFT BREAK will be to load and run the BASIC program automatically.

*HELP <keyword>

Purpose
Displays useful information on the screen. In the disc system this consists of a list of the filing system commands or the utilities depending on the <keyword> used. The two keywords which produce a response in the Disc Filing System are UTILS and DFS. If just *HELP is typed, the system produces a list of currently installed ROMs and a list of the keywords which will produce further information.

Example

```
*HELP

DFS 1.00
   DFS
   UTILS

OS 1.20

*HELP DFS
DFS 1.00

   ACCESS <afsp> (L)
   BACKUP <source> <dest.>
   COMPACT (<drive>)
   COPY <source> <dest.> <afsp>
   DELETE <fsp>
   DESTROY <afsp>
   DIR (<dir>)
   DRIVE <drive>
   ENABLE
   INFO <afsp>
   LIB (<dir>)
   RENAME <old fsp> <new fsp>
   TITLE <title>
   WIPE <afsp>

OS 1.20
```

```
*HELP UTILS

:DFS 1.00
    BUILD <fsp>
    DISC
    DUMP <fsp>
    LIST <fsp>
    TYPE <fsp>
```

Notes

*RUN, *SPOOL, *SAVE, *EXEC and *LOAD are not included in these lists because they are Machine Operating System commands which operate outside the Disc Filing System. *HELP is a Machine Operating System command.

*INFO <afsp>

Purpose
Displays information about a file or group of files. It includes details not given by *CAT such as the length of the file and its location. It is displayed in the following order across the screen.

Directory	File name	Access	Load address	Execution address	Length in bytes	Start sector

Example

```
*INFO A. HELLO
```

displays

```
A.     HELLO L   001900 00801F 00003B 003
```

Description
Displays detailed file information.

Associated commands

*CAT

Notes
If the file is not found on the specified (or assumed) drive and directory the message

```
Not found
```

is produced. The command must be re-entered using the correct <afsp>.

The wild card facilities # and * may be used if you want information about a group of files.

*LIB :(<drv>) <dir> Selects the library

Purpose
Sets the library to the specified drive and directory.

Example

```
*LIB  :1.A
```

sets the library to drive 1 directory A. After this typing

```
*<filename>
```

will search directory A on drive 1 for the named file and if it is found the file will be loaded and executed just as if you had typed

```
*RUN  :1.A.  <filename>
```

Description
Sets the drive/directory containing the library.

Associated commands

*RUN

Notes
The library can contain files which are utility programs, designed to act on other files eg sorts, edits and merges are all common utility programs. It is then possible to say:

```
*SORT  FRED
```

where SORT is the name of the file in the library and FRED is the name of another file in the current drive and directory. This makes use of the fact that any text after the <fsp> is stored in memory and is available to your machine code program for interpretation. A pointer to the start address of this text is available to your program via a call to OSARGS with Y=Ø, A=1 and X=the address of the byte block in page Ø where the text is stored. To read the text stored at this location you must use a call to OSWORD with A=5.

OSWORD call with A=5

Read I/O processor memory. This call enables any program to read a byte in the I/O processor no matter in which processor the program is executing. On entry, X and Y point to a block of memory as follows:

XY LSB of address to be read
XY+1
XY+2
XY+3 MSB of address to be read

On exit, the 8-bit byte will be stored in XY+4.

As this routine reads one byte at a time, you may need to use repeated calls to it to recover all the text following the <fsp>.

Chapter 9 provides more information about using the disc system from assembler programs.

*LIST <fsp>

Purpose
Displays a text file on the screen with line numbers.

Example

```
*LIST DATA
```

displays the contents of the file called DATA on the screen, line by line with each line numbered.

Description
ASCII list with line numbers.

Associated commands

*TYPE
*DUMP

Notes
BASIC is tokenised, so listing a BASIC program file will display nonsense. (See the *BBC Microcomputer User Guide.*) An ASCII text file of a BASIC program can be obtained using the *SPOOL command.

Files written with BASIC keyword PRINT# can also be listed with this command.

In page mode the listing will stop after displaying each screenful, until you press either SHIFT key to make it continue. CTRL N turns page mode on, CTRL O turns it off.

*LOAD <fsp> <address>

Purpose
Reads a named file from the disc into memory in the computer starting at either a specified start address or the file's own load address.

Example

```
*LOAD HELLO
```

Reads the file HELLO into memory starting at location 19ØØ (hex), which is the load address of the file when it was saved.

(See example in *INFO.)

```
*LOAD HELLO 32ØØ
```

Reads the file HELLO into memory starting at location 32ØØ (hex). Other examples are

```
*LOAD "HELLO"
*LOAD "HELLO" 32ØØ
```

Description
Loads a file into memory.

Associated commands

*SAVE
*RUN

Notes
All the above are valid commands. The quotation marks are optional; but either a pair or none should be present. The named file must be in the current directory on the current disc. If the file is not found, the message

```
Not found
```

is produced.

*OPT 1 (n)

Purpose
This command enables or disables a message system which displays a file's information (the same as *INFO). Every time a file on the disc is accessed, the information is displayed. (n) can be any number greater than Ø to enable the feature. (n) = Ø disables it.

Example

```
*OPT 1 1 or *OPT 1, 1
```

enables the messages;

```
*OPT 1 Ø or *OPT 1, Ø
```

disables the messages.

Description
Message system to display file information at every access.

Associated commands

*INFO

Notes
A space or a comma between *OPT 1 and its argument (n) is essential.

*OPT 4 (n)

Purpose
Changes the auto-start option of the disc in the currently selected drive. There are four options to choose from: Ø, 1, 2 or 3. Each option initiates a different action when you press SHIFT and BREAK on the computer. The computer will either ignore or automatically *LOAD, *RUN or *EXEC a file called !BOOT which must be in the directory $ on drive Ø.

Example

```
*OPT  4 Ø   does nothing
*OPT  4 1   will *LOAD the file !BOOT (ie a machine-code file)
*OPT  4 2   will *RUN the file !BOOT (ie a machine-code file)
*OPT  4 3   will *EXEC the file !BOOT (ie a BASIC file)
```

When *OPT 4 2 is set, and a machine-code program (ie !BOOT) is executed, it is important to remember that the interrupt flag is undefined (unlike *RUNning the program, where the interrupt flag is cleared). It is suggested that your machine code !BOOT files contain a CLI to clear the interrupt flag.

Description
Changes the start-up option of a disc.

Notes
It is essential to include a space between the command and (n).
*OPT4Ø would produce the message

```
Bad option
```

If the disc is write-protected the error message

```
Disk read only
```

is produced in response to the *OPT 4 command.

If the option Ø is set, the !BOOT file need not be there. With any other option the message

```
Not found or File not found
```

is produced if !BOOT is not found in directory $ on drive Ø.

Important: Do not confuse *OPT 4 with BASIC keyword OPT or *OPT 1. They are completely different. Refer also to chapter 12 where it describes how to swap the effects of BREAK and SHIFT BREAK.

The utilities disc is set to option 2 when you receive it so when you press SHIFT BREAK the 'HELLO' program, saved as file !BOOT, is *RUN automatically.

*RENAME (old fsp) (new fsp)

Purpose
Changes the file name and moves it to another directory if required.

Example

```
*RENAME  SUMS  B.MATHS
```

Assuming that the current directory is $, the file `$.SUMS` becomes `B.MATHS`.

Description
Renames a file.

Notes

```
*RENAME   :0.$.SUMS   : 1.B.MATHS
```

is not allowed. The file cannot be moved from drive Ø to drive 1 using `*RENAME`. Only the directory and file name can be changed.

If the file does not exist the message

```
Not found
```

is displayed. If the first file is locked

```
Locked
```

is displayed. If the disc is write-protected

```
Disk read only
```

is displayed. If the <new fsp> has already been used the message

```
Exists
```

is displayed.

*RUN <fsp> (parameters to utility)

Purpose
This command is used to run machine-code programs. It loads a file into memory and then jumps to the execution address of that file.

Example

```
*RUN   PROG
```

will cause a machine-code program in the file called PROG to be loaded and executed starting at the execution address of the file.

Description
Runs a machine-code program.

Associated commands

*SAVE
*LIB (for an explanation of 'parameters to utility')

Notes
This command will not run a BASIC program.

Typing *<fsp> is accepted as being *RUN <fsp>.

Typing *<filename> results in the file being loaded and executed if it is found in the currently selected drive/directory or the library.

*SAVE <filename> <start address> <finish address> (<execute addr.>) (<reload addr.>)

Purpose
It is important not to confuse this with the BASIC keyword SAVE – they are quite different. This command takes a copy of a specified section of the computer's memory and writes it on to the disc in the current drive/directory. It is put into a file of the given name. You will mostly use this command to record your machine-code programs.

Example

```
*SAVE "PROG"  SSSS FFFF (EEEE) (RRRR)
*SAVE "PROG"  SSSS + LLLL (EEEE) (RRRR)
```

SSSS = Start address of memory to be saved
FFFF = Finish address
EEEE = Execution address (see below)
RRRR = Reload address
LLLL = Length of information

Notes
RRRR and EEEE may be omitted in which case the reload address and the execution address are assumed to be the same as the start address.

If the disc is write-protected the message

```
Disk read only
```

is produced. If there are already 31 files on the disc the message

```
Cat full
```

is produced. If the specified file name already exists and is locked the message

```
Locked
```

is produced. If the file already exists but is unlocked it is deleted. Then starting in sector 2 track Ø a gap large enough to hold the new information is searched for. If none is found the message

```
Disk full
```

is produced.

If enough space is available the information is written on to the disc and the file name is entered on to the catalogue in the current directory.

*SPOOL <fsp>

Purpose
Prepares a file of the specified name on the disc to receive all the information subsequently displayed on the screen. This is a very useful command particularly for producing a text file of one of your BASIC programs. (See notes below.)

Example
You can obtain a text file of one of your BASIC programs as follows:

```
LOAD   "MYPROG"
```

Loads a program from disc into memory.

```
*SPOOL  FRED
```

Opens a file called FRED on the disc ready to receive information from the screen.

```
LIST
```

Causes the BASIC program to be displayed on the screen and also written onto the file called FRED.

```
*SPOOL
```

Turns off the 'spooling' and closes the file called FRED.

Description
Spools subsequent output to the screen to a named file opened for the purpose. Closes the file when spooling is terminated.

Associated commands

*BUILD
*EXEC
*LIST
*TYPE

Notes

BASIC on the BBC Microcomputer is 'tokenised'. This means that the lines which you type in your program are abbreviated inside the computer's memory and on the disc. A program file will contain these abbreviated 'tokens' rather than your original program text.

*TITLE <disc name>

Purpose
Changes the title of the disc in the current drive to the first 12 characters after the command. It fills in with spaces if there are less than 12 characters. Any characters are allowed.

Example

```
*TITLE "MY DISC"
```

This sets the title to `"MY DISC"` with five spaces added on the end.

```
*TITLE "A DIFFERENT TITLE"
```

This changes the title to `A DIFFERENT`. Anything after the first 12 characters is ignored. The quotation marks are only required if the title includes spaces.

Notes
If the disc is write-protected the message

```
Disk read only
```

appears when you try to use this command.

*TYPE <fsp>

Purpose
Displays a text file on the screen without line numbers.

Example

```
*TYPE HELLO
```

Description
Screen list of a named file.

Associated commands

```
*LIST
*DUMP
```

Notes
BASIC programs are not stored on disc as text files when you SAVE them, so this command will display nonsense if you try to display a BASIC program.

Page mode is selected with CTRL N and turned off by CTRL O. Page mode allows you to see a screenful of text at a time: when the screen has filled up, instead of immediately scrolling, the computer waits until you press the SHIFT key. The next screenful then appears.

*WIPE <fsp>

Purpose
Removes specified files from the catalogue and rearranges the catalogue. Asks for confirmation that each file conforming to the specification is to be deleted.

Example

```
*WIPE  *.SU*
```

is a request to delete all files on the current drive beginning with the letters SU. As each file is found the file name is displayed like this:

```
A.SUM          :
```

At this point only type Y if you want to delete the file. Typing anything else leaves the file intact.

Description
Delete with <afsp> and confirmation per file.

Associated commands

```
*DESTROY
*DELETE
```

Notes
Once deleted using *WIPE, a file cannot be restored. Locked files are not removed. (See *ACCESS.)

7 The filing system utilities

As explained in chapter 6 the filing system commands are, in fact, programs stored in ROM (Read Only Memory) inside the BBC Microcomputer.

The utilities are similar programs but they are stored on the utilities disc. They are used in the same way as the commands by typing

*(utility name)

The utilities disc must be in the disc unit at the time. The utility must either be in the current directory on the current drive or in the library (see command *LIB).

There are two 'formatting' utilities supplied: one for 40 track discs, and the the other for 80 track discs. On the utilities disc these are normally in drive 0, directory $, which are the default values on start-up for both the library and the current drive/directory. (See commands *DRIVE and *DIR.) The third utility is *VERIFY.

The following pages describe the utilities. You can add utilities of your own by saving machine-code programs into the library.

*FORM40 (<drv>)

Format a 40 track disc

Purpose

The command prepares new discs for use with the filing system on the BBC Microcomputer. It marks areas of disc where information will be stored and sets up a catalogue. The catalogue is empty at first, but when you store programs and data on the disc, the catalogue records their positions on the disc. They can then be retrieved quickly by reference to the catalogue. While formatting the information put on to the disc is verified automatically.

Example

To format a new disc insert the utilities disc into the disc drive and type

```
*FORM40
```

The computer asks

```
Do you really want to format drive 0?
```

Remove the utilities disc. Insert the disc to be formatted and then answer

```
Y
```

The formatting starts immediately and the message

```
Formatting drive (n)
```

is displayed. As each track is formatted and verified the track number is displayed as follows:

```
00 01 02 03 04 05 06 07 08 09
0A 0B 0C 0D 0E 0F 10 11 12 13
14 15 16 17 18 19 1A 1B 1C 1D
1E 1F 20 21 22 23 24 25 26 27
disc formatted
```

Above is the response if the formatting was successful. If a ? appears in

this list, it means that a re-try occurred, so you're advised to reformat the disc. If the formatting was not successful, then either the message

```
Verify error
```

or

```
Format error
```

is displayed.

After successful formatting the message

```
Repeat format (Y/N)
```

is displayed. You can format another disc straight away.

Description
Initialises discs with track and sector format. Clears the catalogue and verifies the sectoring.

Associated commands

```
*FORM80
```

Notes
*FORM80 is the same except that it formats the 80 track discs used in the dual drive system.

If you find that verification or formatting fails persistently it could be that either the discs you have are poor quality or that the disc drive needs servicing.

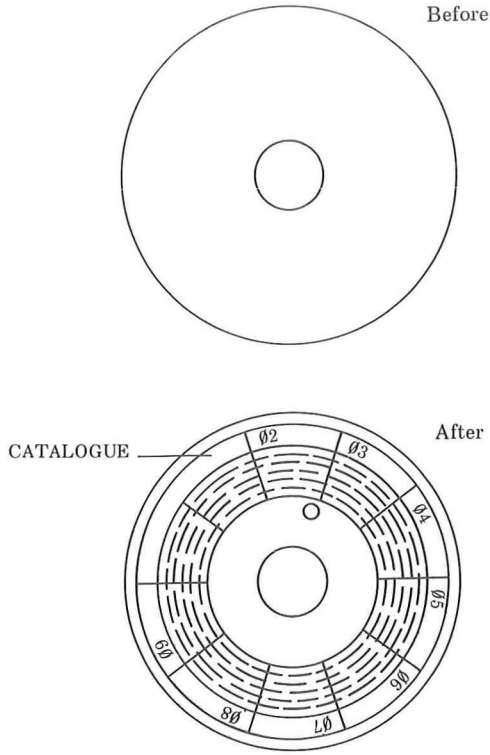
If you don't specify the drive number, the system will prompt with

```
Format which drive?
```

You can enter 0, 1, 2 or 3.

If you have a dual drive system you can format a disc in the second drive while the utilities disc remains in the first drive.

Fig 7 Formatting



*VERIFY (<drv>)

Purpose
Checks each sector of a disc for legibility. It is done automatically when you use the *FORM40 and *FORM80 commands.

Example

```
*VERIFY 1
```

verifies drive 1.

Description
Sector verification program.

Associated commands

*FORM40
*FORM80

8 Random access files

One of the major advantages of a disc over a cassette tape is that the read/write head of the disc drive can be moved to a specific place on the disc quickly and accurately. Imagine you have a data file on cassette tape consisting of 'Names' and 'Telephone numbers'. To find a specific telephone number the file must be loaded and read from the beginning until the required record is found. If the file is long this will take some time. On the other hand, the Disc Filing System allows you to move to the required record and just read that one. Clearly this is much quicker.

To make this possible the Disc Filing System provides a pointer. The pointer points to a particular character in the file. It is the next character on the file to be read or written. In BASIC the pointer is controlled by the keyword PTR#. The other keywords in BASIC which are used in connection with disc files are EXT# and EOF#. EXT# tells you how long a file is, EOF# returns a value of TRUE (-1) if the end of file has been reached and FALSE (0) if not. All the BASIC keywords used to manipulate disc files are explained in the User Guide. They are:

```
OPENOUT
OPENIN
PTR#
EXT#
INPUT#
PRINT#
BGET#
CLOSE#
BPUT#
EOF#
```

To prepare a file to receive data the OPENOUT keyword is used. In the User Guide the following example is given:

```
330  X=OPENOUT ("cinemas")
```

The effect of this line in a BASIC program is as follows:

1. If a file called 'cinemas' exists it is deleted.

2. A file called 'cinemas' is entered on to the disc catalogue of the currently selected drive, in the current directory.

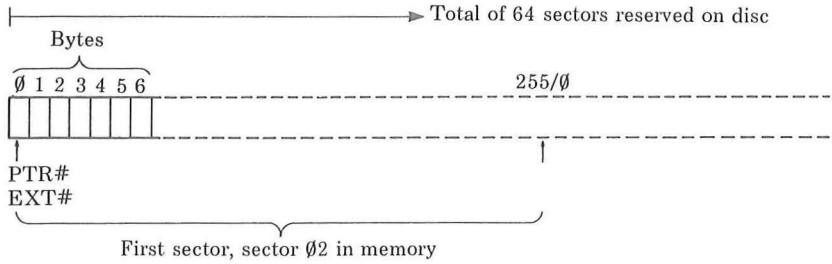
3. The filing system reserves 64 sectors (or the length of the previous file called 'cinemas' if there was one) on the disc for the exclusive use of the file 'cinemas'. If 64 sectors are not available, the file is not created and an error is produced.

4. Evaluating PTR# and EXT# at this point will reveal that they are both set to zero.

5. The filing system will have loaded into memory the first sector, 256 bytes of the file. This area of memory is specially reserved by the filing system for this purpose and is referred to as the 'buffer'.

Notice that the first action of the keyword OPENOUT is to delete any existing file of the specified name.

If there were no files on the disc previously, the effect can be illustrated as follows:

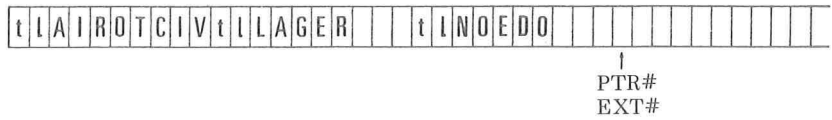


Nothing has been written on the file, so the value of EXT# (extent) is zero.

We can now use the BASIC keyword PRINT# to write three cinema names into these slots of ten characters each, as follows:

```
340  A = PTR#X
350  PRINT# X, "VICTORIA"
360  PTR#X = A+1Ø
370  PRINT# X,"REGAL"
380  PTR# X = A+2Ø
390  PRINT# X, "ODEON"
400  PTR# X = A+3Ø
```

In practice you can do it much more elegantly than shown above. Nevertheless the result immediately after line 4ØØ is:



Notice that the cinema names (in this illustration, VICTORIA) are in the file backwards. They are preceded by two bytes, represented in the diagram by 't' and 'l' . 't' specifies the type of data which follows. In this case the type is 'string' so the first byte will contain &ØØ in hex, as indicated in the table below.

't' = &ØØ = String type, followed by 'l', followed by the string.

't' = &4Ø = Integer type, followed by four bytes containing the integer.

't' = &FF = Real type, followed by five bytes containing the real number.

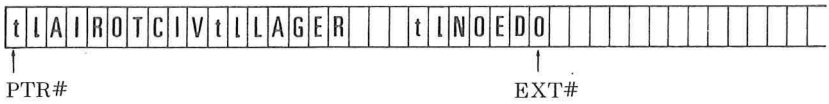
In our example the second byte, represented by 'l', gives the length of the string in hex. The integer and real number types are of fixed length as indicated above so they do not require the byte represented by 'l' to give the length. Real numbers are stored in exponential format, integers are stored with the high order bytes first in the file.

In the example we have used only the first 26 bytes of the file, so everything written to the file fits into the first sector which is in a 'buffer' in memory. If we had gone on writing names, the filing system would eventually have put the information in the memory buffer on to sector Ø2 of the disc and loaded sector Ø3 into the buffer to continue. This is still assuming that there are no other files on the disc, otherwise different sectors would be used. Remember that sectors ØØ and Ø1 are reserved by the Disc Filing System for the disc catalogue. Clearly then, at the end of a sequence of writing actions, we are left with a buffer in memory which may be partly filled with information. We must make sure that this information is written to the disc. This is done with the `CLOSE#` keyword in BASIC. This empties the buffer and frees the channel on which we opened the file (X in the example).

We can now read the information back from the disc if we want to. OPENIN is the BASIC keyword used to do this, eg:

```
5   DIM cine$(3)
10  X = OPENIN ("cinemas")
20  B = 1
30  FOR A = 0 TO 20 STEP 10
40  PTR#X = A
50  INPUT#X,cine$(B)
60  B = B+1
70  NEXT A
```

Line 10 of the example opens the file 'cinemas', loads the first sector into the buffer and sets PTR# to zero and EXT# to the length of the file.



Lines 30 to 50 of the example read each cinema name into an element of the array cine$, advancing the pointer to the start of the next name after reading each one. Now you can see why we stored each name in its own '10-byte record'. This makes it much easier to write a program to find the names again.

The important principle about using random access files is that you must keep track of where each item of information is written. You can then set PTR# to point to it again when you want to read or change it. The examples illustrate the basis of a very simple technique. There are a number of others which you can devise.

Note 1: As shown earlier in this discussion, OPENOUT reserves 64 sectors for a file. Other files opened may reserve sectors which immediately follow, eg:

```
X = OPENOUT ("cinemas")
Y = OPENOUT ("clubs")
```

The statements reserve 128 sectors consecutively if the disc was otherwise empty. It may be that you require more than 64 sectors for the first

file 'cinemas'. If so, you will need to write 'dummy' records to the file to extend it to the required length before you open the second file, eg:

```
10  X = OPENOUT ("cinemas")
20  FOR   A = 1 TO 200
30  PRINT#X, "DUMMY NAME FIELD"
40  PRINT#X, "DUMMY ADDRESS LINE ONE"
50  PRINT#X, "DUMMY ADDRESS LINE TWO"
60  PRINT#X, "ADDRESS LINE THREE"
70  PRINT#X, "POST CODE 123"
80  NEXT A
```

This program creates a file 79 sectors long with the dummy name and address written every 100 bytes.

By writing beyond the reserved area in this way you can effectively reserve as many sectors as you like. You can then open other files in the remaining space on the disc. `EXT#` will give the position of the '3' after the last dummy address on the file (20000).

The above method will only work consistently if you start with a blank, formatted disc. If you want to create a random access file larger than 64 sectors on a disc with other files already on it, there is another method.

First save the file with the name you want and with the number of bytes required. Use the address parameters of the `*SAVE` command to specify the number of bytes, eg:

```
*SAVE "DATA" 00000 08000
```

will create a file of 128 sectors (32K) called `DATA`. You can then open the file later in your program. The file will contain miscellaneous data which you can overwrite with the information you actually want. This second method causes the filing system to search the disc for a free space large enough to hold the file. Existing files will be skipped over if they would otherwise overlap with the new file.

Note 2: Up to five files may be open at any one time. This is because the space reserved for each file in the computer's memory to hold the information about extent, pointer etc is limited. In certain versions of the Disc Filing System, the commands `*LOAD`, `*SAVE`, `*EXEC`, `*SPOOL`, `*BUILD`, `*LIST` and `*DUMP` each use the space occupied by the information relating to one open file while they are active.

9 Using the filing system in assembler

Section 43 of the new User Guide is essential reading for anyone wanting to write assembler programs on the BBC Microcomputer. Most of the necessary information for using the filing system in assembler is presented there. In this chapter the main points are summarised and a particular use of OSWORD is described in detail.

General principles

There are a number of routines available to handle disc I/O. All the routines must be called with a JSR and the decimal flag clear. It is important that you use these routines. They are called in address range &FFØØ to &FFFF. They then call an internal (ROM resident) routine whose address is stored in RAM between &Ø2ØØ and &Ø2FF. The address here will vary according to the filing system in use. For example, the routine OSFIND to open or close a file is entered at &FFCE. It is indirected via &Ø21C. &Ø21C and &Ø21D contain the address of the executable routine in the Disc Filing System ROM. You can intercept the call by changing the addresses in these RAM locations.

Using the available routines you can perform all necessary functions relating to disc files. The relevant routines together with their entry points are as follows:

OSFIND	&FFCE	FINDV	&Ø21C	Open or close a file
OSARGS	&FFDA	ARGSV	&Ø214	Load or save data about a file
OSFILE	&FFDD	FILEV	&Ø212	Load or save a complete file
OSBGET	&FFD7	BGETV	&Ø216	Read a single byte to A from the file
OSBPUT	&FFD4	BPUTV	&Ø218	Write a single byte from A to the file
OSGBPB	&FFD1	GBPBV	&Ø21A	Load or save a number of bytes
OSWORD	&FFF1	WORDV	&Ø2ØC	With A=&7F and a parameter block, loads or saves a sector

OSFIND

Opens a file for writing or reading and writing. The routine is entered at &FFCE and indirects via &21C. The value in A determines the type of operation.

A=Ø	Causes a file or files to be closed
A=&4Ø	Causes a file to be open for input (reading)
A=&8Ø	Causes a file to be opened for output (writing)
A=&CØ	Causes a file to be opened for input and output (random access)

If A=&4Ø, &8Ø or &CØ the Y (high byte) and X (low byte) must contain the address of locations in memory which contain the file name terminated with carriage return (&ØD). On exit A will contain the channel number allocated to the file for all future operations. If A=Ø on exit, then the operating system was unable to open the file.

If A=Ø then a file, or all files, will be closed depending on the value of Y. Y=Ø will close all files, otherwise the file whose channel number is given in Y will be closed.

On exit C, N, V and Z are undefined and D=Ø. The interrupt state is preserved, but interrupts may be enabled during the operation.

OSARGS

This routine enables a file's attributes to be read from file or written to file. The routine is entered at &FFDA and indirects via &214. On entry, X must point to four locations in zero page and Y contains the channel number.

If Y is non-zero, then A will determine the function to be carried out on the file whose channel number is in Y.

A = 1	Write sequential pointer
A = Ø	Read sequential pointer
A = 2	Read length
A = &FF	'Ensure' that this file is up to date on the media

If Y is zero then the contents of A will determine the function to be carried out.

A=Ø will return, in A, the type of file system in use. The value of A on exit has the following significance.

Ø No file system
1 12ØØ baud cassette file system
2 3ØØ baud cassette file system
3 ROM pack file system
4 Disc file system
5 Econet file system
6 Teletext/Prestel Telesoftware file system

A=1 will return the address of the rest of the command line in the zero page locations.

A=&FF will 'ensure' that all open files are up to date on the media.

On exit
X and Y are preserved, C, N, V and Z are undefined and D=Ø. The interrupt state is preserved, but interrupts may be enabled during the operation.

OSFILE

This routine, by itself, allows a whole file to be loaded or saved. The routine is entered at &FFDD and indirects via &212.

On entry
A indicates the function to be performed. X and Y point to an 18-byte control block anywhere in memory. X contains the low byte of the control block address and Y the high byte. The control block is structured as follows from the base address given by X and Y.

OSFILE control block

ØØ Ø1	Address of file name, which must be terminated by &ØD	LSB MSB
Ø2 Ø3 Ø4 Ø5	Load address of file	LSB MSB

Ø6 Ø7 Ø8 Ø9	Execution address of file	LSB MSB
ØA ØB ØC ØD	Start address of data for write operations or length of file for read operations	LSB MSB
ØE ØF 1Ø 11	End address of data, ie byte after last byte of data to be written, or file attributes	LSB MSB

The following table indicates the function performed by OSFILE for each value of A.

A=Ø	Save a section of memory as a named file. The file's catalogue information is also written
A=1	Write the catalogue information for the named file
A=2	Write the load address (only) for the named file
A=3	Write the execution address (only) for the named file
A=4	Write the attributes (only) for the named file
A=5	Read the named file's catalogue information; place the file type in A
A=6	Delete the named file
A=&FF	Load the named file

When loading a file, the byte at XY+6 (the LSB of the execution address) determines where the file will be loaded in memory. If it is zero then the file will be loaded to the address given in the control block. If non-zero then the file will be loaded to the address stored with the file when it was created.

The file attributes are stored in four bytes. The least significant eight bits have the following meanings:

Bit	**Meaning**
Ø	Not readable by you
1	Not writable by you

2	Not executable by you
3	Not deletable by you
4	Not readable by others
5	Not writable by others
6	Not executable by others
7	Not deletable by others

File types are as follows:

Ø	Nothing found
1	File found
2	Directory found

A BRK will occur in the event of an error, and this can be trapped if required.

On exit X and Y are preserved, C, N, V and Z are undefined and D=Ø. The interrupt state is preserved, but interrupts may be enabled during the operation.

Read/write one byte

OSBGET call address &ØFFD7 to get a byte.
OSBPUT call address &ØFFD4 to put a byte.

Y contains the channel number on which the file was opened using OSFIND.
X is not used, but preserved.
A contains the byte to be put or receives the byte which is read.

The position in the file where the action of the GET or PUT takes place is determined by the position of the pointer as set by OSARGS.

On exit
C clear implies a successfully completed transfer.
C set implies end of file reached before completion of transfer.

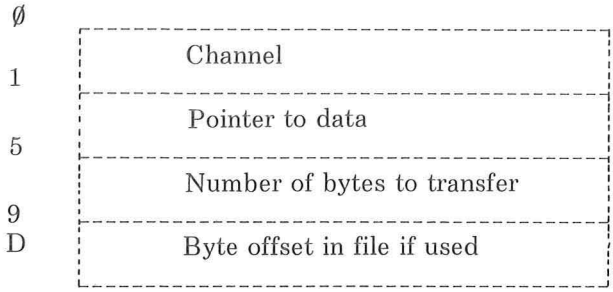
Read/write a group of bytes

OSGBPB call address &ØFFD1

This routine will write or read a byte (or group of bytes) to or from a specified open file. The option to read or write is determined by the value of A. The length of the data and its location are specified in a control block in memory.

On entry
X (low byte) and Y (high byte) point to the instruction block:

Offset



A determines the type of operation:

A = &Ø1 Put byte using byte offset
A = &Ø2 Put byte ignoring byte offset
A = &Ø3 Get byte using byte offset
A = &Ø4 Get byte ignoring byte offset
A = &Ø5 Read title, option, drive
A = &Ø6 Read current directory
A = &Ø7 Read current library
A = &Ø8 Read file names

This method is particularly useful in the environment of the Econet® file system, where the 'packaging overhead' for transferring small amounts of data is proportionately high.

On exit
A = Ø implies operation attempted.
A unchanged implies filing system does not support this facility.
C clear implies a successfully completed transfer.
C set implies end of file reached before completion of transfer. The number of bytes and byte offset (if used) are modified to show how much data has been transferred (usually as much as was asked for) and the new pointer value is the old pointer plus the amount transferred.

A = &Ø1 to &Ø4
The number of bytes is modified to show how much data has not been transferred (usually zero). The pointer value is updated (ie old pointer

Econet is a trademark of Acorn Computers Limited.

plus the amount transferred), and the offset is set to its current value.

A = &Ø5
This returns in the area pointed at by the pointer the title, option and drive of the currently selected disc. It is returned in this form:

<title length><title><option><drive>

The cycle number, option and drive are single binary bytes.

A = &Ø6
This returns the currently selected directory. It is returned in this form:

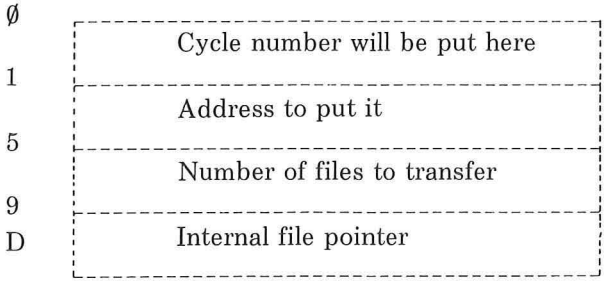
<length of disc name><disc name><length of directory name><directory name>

A = &Ø7
This returns the currently selected library. It is returned in this form:

<length of disc name><disc name><length of library name><library name>

A = &Ø8
This returns the files in the current directory. The format of the control block is similar to that for sequential files:

Offset



If the pointer is set to zero, the search will begin with the first file. All are updated in a similar manner to the way pointers are updated for A=&Ø1 to &Ø4. The format of the file names is as follows:

<length of filename 1><filename 1><length of filename 2><filename 2>. . .

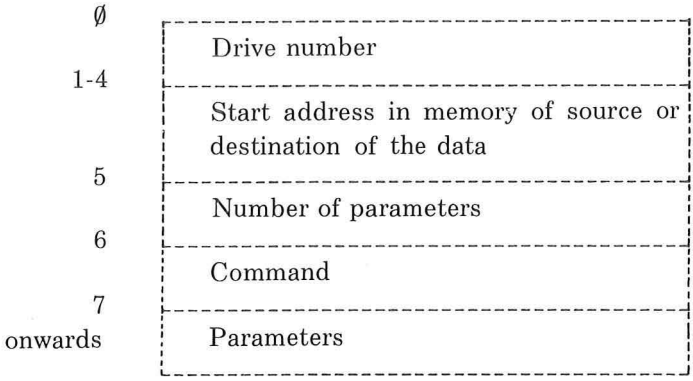
Read/write a sector

OSWORD with A =&7F
Call address at &FFF1

A=&7F indicates that a general read/write operation is required.

On entry X (low byte) and Y (high byte) point to the instruction block:

Offset



Example:

Number of parameters = 3

Command =&53 to read or &4B to write

Parameter 1	=	Track number
Parameter 2	=	Sector number
Parameter 3	=	&21 (specifies sector length of 256 bytes and 1 to be acted upon)

On exit
Ø in the last parameter address +1 indicates a successful transfer. A failure is indicated by a disc error number.

1Ø Changing filing systems

Your computer can have several filing systems available other than the Disc Filing System. The following commands are all used to exit from the current filing system into the one named.

*TAPE3	3ØØ baud cassette
*TAPE12	12ØØ baud cassette
*TAPE	12ØØ baud cassette
*NET	Econet filing system
*TELESOFT	The Prestel and Teletext system
*ROM	The cartridge ROM system
*DISC	Enters the filing system from one of the others
*DISK	Alternative spelling for above

Typing the command to enter the system you are already in has no effect.

11 Error messages

&BD Not enabled
An attempt is made to use a restricted command without typing *ENABLE immediately preceding it.

&BE Cat full
The catalogue has enough space for 31 files. This error is produced if you attempt to enter more than 31.

&BF Can't extend
An attempt is made to extend a random access file when there is insufficient space immediately after it to do so.

&CØ Too many open
This error occurs on attempting to open a sixth random access file. Five is the maximum allowed at once.

&C1 Read only
This error occurs if you try to write to a file opened for reading only, using OSFIND with A=&4Ø.

&C2 Open
This error occurs if you attempt to open a file which is already open. An intervening CLOSE is required between two OPENs referring to the same file. This error is also produced if you try to delete an open file with the commands: *DELETE, *SAVE, *BUILD etc.

&C3 Locked
Access to the named file is locked. This error message is displayed when any attempt is made to overwrite or write to a locked file.

&C4 Exists
This occurs if you try to rename a file with an existing file name.

&C5 Drive fault
This error means that the disc drive is probably faulty and needs attention.

&C6 Disk full
This indicates that there is not enough space on the disc to open (OPENOUT#) or save a file of the specified size.

&C7 Disk fault
If this error message occurs it means that the computer cannot read the disc. It implies that the disc is damaged, faulty, unformatted or of the wrong type, like an 8Ø track disc in a 4Ø track drive.

&C8 Disk changed
This error occurs if the computer detects that a disc has been changed while files on it are still open.

&C9 Disk read only
This error occurs if you attempt to write to a disc which has the write-protection notch covered.

&CB Bad option
There are currently two 'option' commands *OPT1 and *OPT4. The error occurs if you type anything else besides 1 and 4 after *OPT.

&CC Bad name
This message appears if you enter a file name which is invalid, such as: longer than seven characters, or a blank file name, or control characters.

&CD Bad drive
This error means that the :(drv) part of the file specification was incorrect, eg ':' colon missing or drive number out of range Ø to 3.

&CE Bad dir
Means that the specified directory is not allowed, eg more than one character.

&CF Bad attribute
This error occurs if you use anything other than the letter 'L' with the *ACCESS command.

&D6 Not found
The Disc Filing System could not find the named file in the specified drive/directories.

&DC Syntax
You have typed in a command with bad arguments such as *COPY RETURN. The error message includes the correct syntax.

&FE Bad command
This means that * was ommitted or that the command name was not recognised as a Disc Filing System command or utility.

12 Technical information

18-bit addressing

The BBC Disc Filing System uses 18-bit addressing giving a range from &ØØØØØ to &3FFFF. This means that two bytes and two bits are required to store a complete address, like this:

```
&3FFFF is     11   1111 1111     1111 1111     in binary
            High    Middle          Low
            bits    bits            bits
```

Each full address therefore consists of high, middle and low order bits. This is important to note, because the bits of the address are not always stored consecutively in the catalogue. This is clearly shown by the way that the disc catalogue is loaded into memory.

Another important factor concerns the use of a second processor. If the top two bits of an address are set, eg: &3----, the address is assumed to refer to the I/O processor.

Disc catalogue

Sectors ØØ and Ø1 on the disc are reserved for the catalogue. The format of the catalogue is as follows:

Sector ØØ

&ØØ to &Ø7	First eight bytes of the 13-byte disc title
&Ø8 to &ØE	First file name
&ØF	Directory of first file name
&1Ø to &1E	Second file name
&1F	Directory of second file name
	. . . and so on

Repeated up to 31 files.

Sector Ø1

&ØØ to &Ø3	Last four bytes of the disc title
&Ø4	Sequence number
&Ø5	The number of catalogue entries multiplied by 8
&Ø6 (bits Ø,1)	Number of sectors on disc (two high order bits of 1Ø bit number)
(bits 4,5)	!BOOT start-up option
&Ø7	Number of sectors on disc (eight low order bits of 1Ø bit number)
&Ø8	First file's load address, low order bits
&Ø9	First file's load address, middle order bits
&ØA	First file's exec address, low order bits
&ØB	First file's exec address, middle order bits
&ØC	First file's length in bytes, low order bits
&ØD	First file's length in bytes, middle order bits
&ØE (bits Ø,1)	First file's start sector, two high order bits of 1Ø bit number
(bits 2,3)	First file's load address, high order bits
(bits 4,5)	First file's length in bytes, high order bits
(bits 6,7)	First file's exec address, high order bits
&ØF	First file's start sector, eight low order bits of 1Ø bit number
	. . . and so on

Repeated for up to 31 files.

Note that the first eight bytes in each sector contain miscellaneous information about the disc. Each subsequent block of eight bytes contains information about the files, repeated for up to 31 files. The complete information about file 3 would be found in the fourth block of eight bytes on sector ØØ followed by the fourth block of eight bytes on sector Ø1.

File system initialise and !BOOT

On pressing **BREAK**, the MOS (Machine Operating System) seeks and initialises a filing system. It starts with the service ROM closest to the edge of the main printed circuit board. The first one will be initialised if just **BREAK** was pressed.

If another key is held down while you press **BREAK**, the first file system which recognises the key will be initialised. If none recognises the key, the CFS (cassette filing system) is initialised.

Having initialised a filing system, if 'link 5' is unmade (see below) or if SHIFT is not pressed and 'link 5' is made, then the start-up option of any associated device is examined. For options 1 to 3, the Disc Filing System will search for a file named !BOOT on the device and act according to the start-up option set.

In the case of the cassette filing system the start-up option of the ROM cartridge is examined as no start-up options are provided on cassette tapes.

Link facilities

Under the top cover of the computer to the right of the space bar, you will find eight pairs of holes in the circuit board. These are provided for the fitting of links, and give you some extra facilities, described in detail below. In the following tables, a zero means that a link is not made, and a one means that a link is made.

Links 1 and 2
Reserved for future expansion.

Links 3 and 4
The table below shows the optimum link settings for different types of disc drives which may be used with the BBC Microcomputer.

Drive type	Step	Settle	Headload	Links 3 and 4	
Tandon 8Ø track	4mS	16mS	Ø	1	1
Tandon 4Ø track, Shugart	6mS	16mS	Ø	1	Ø
Teac, Mitsubishi, Olivetti 8Ø track, Siemens 211, 221, YD28Ø	6ms	16mS	56	Ø	1
Olivetti 4Ø track, Shugart SA2ØØ, BASF 61Ø6	24mS	2ØmS	64	Ø	Ø *

If you are using a drive other than those listed above, then obtain the drive speed specification from your supplier. Set the links so that each parameter is greater than or equal to the specification given. After changing the link settings, press CTRL BREAK to simulate a power down reset.

If in doubt, run at the slowest speeds (marked with an asterisk in the above list).

Link 5
See the previous section called 'File system initialise and !BOOT'.

Links 6, 7 and 8
These determine what screen mode the computer starts up in after a cold start (see the *BBC Microcomputer User Guide*).

Screen mode	Links 6, 7 and 8
Mode 7	Ø Ø Ø
Mode 6	Ø Ø 1
Mode 5	Ø 1 Ø
Mode 4	Ø 1 1
Mode 3	1 Ø Ø
Mode 2	1 Ø 1
Mode 1	1 1 Ø

13 Filing system command summary

Command	Abbreviation	
*ACCESS	(*A)	Locks or unlocks a file
*BACKUP	(*BAC.)	Copies all information from one disc to another
*BUILD	(*BU.)	Causes all subsequent keyboard entries to be stored in the named file
*CAT	(*.)	Displays a disc catalogue
*COMPACT	(*COM)	Collects all files on a disc together into one contiguous sequence leaving a single block of free space
*COPY	(*COP.)	Copies all specified files from one disc to another
*DELETE	(*DE.)	Removes a single named file from the disc
*DESTROY	(*DES.)	Removes specified files from a disc in a single action
*DIR	(*DI.)	Changes the current set directory
*DRIVE	(*DR.)	Changes the current set drive
*DUMP	(*DU.)	Produces a hexadecimal listing of a file
*ENABLE	(*EN.)	Allows the use of *BACKUP and *DESTROY
*EXEC	(*E.)	Reads a disc file byte by byte as if the byte were being typed on the keyboard
*HELP	(*H.)	Displays the file system commands with syntax guidelines
*INFO	(*I.)	Displays information about specified files

*LIB	none	Selects the drive/directory for the library
*LIST	none	Displays text file on the screen with line numbers
*LOAD	(*L.)	Reads a file from disc to memory
*OPT1	(*O.1)	Switches the screen messages which accompany disc accesses on or off
*OPT4	(*O.4)	Specifies the auto-start option of a disc, relating to a file named :0/.$.!BOOT
*RENAME	(*RE.)	Changes a file name
*RUN	(*R.)	Runs a machine-code program
*SAVE	(*S.)	Saves a specific part of memory to the disc
*SPOOL	(*SP.)	Transfers all text subsequently displayed on the screen into a specified file
*TITLE	(*TI.)	Changes the title of a disc
*TYPE	(*TY.)	Displays a text file on screen with line numbers
*WIPE	(*W.)	Removes specified files from the disc after confirmation for each file meeting the given specification

Index